HP GlancePlus

For the HP-UX operating system

Software Version: 11.02

Dictionary of Operating System Metrics

Document Release Date: December 2011

Software Release Date: October 2011

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2010 - 2011 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Introduction

This dictionary contains definitions of the Linux operating system performance metrics for HP GlancePlus.

HP GlancePlus provides metrics for system resources, processes, and applications data.You can use the graphical user interface or character-based terminal of HP GlancePlus to view these metrics. This document provides descriptions of each metric. Metrics are arranged in the alphabetical order and grouped by metric classes.

Metric Names by Data Class

Global Metrics

GBL_ACTIVE_CPUGBL_ACTIVE_CPU

GBL_ACTIVE_CPU_COREGBL_ACTIVE_CPU_CORE

GBL_ACTIVE_PROCGBL_ACTIVE_PROC

GBL_ALIVE_PROCGBL_ALIVE_PROC

GBL_BLANKGBL_BLANK

GBL_BOOT_TIMEGBL_BOOT_TIME

GBL_CACHE_QUEUEGBL_CACHE_QUEUE

GBL_CACHE_WAIT_PCTGBL_CACHE_WAIT_PCT

GBL_CACHE_WAIT_TIMEGBL_CACHE_WAIT_TIME

GBL_CDFS_QUEUEGBL_CDFS_QUEUE

GBL_CDFS_WAIT_PCTGBL_CDFS_WAIT_PCT

GBL_CDFS_WAIT_TIMEGBL_CDFS_WAIT_TIME

GBL_COLLECTORGBL_COLLECTOR

GBL_COMPLETED_PROCGBL_COMPLETED_PROC

GBL_CPU_CLOCKGBL_CPU_CLOCK

GBL_CPU_CSWITCH_TIMEGBL_CPU_CSWITCH_TIME

GBL_CPU_CSWITCH_TIME_CUMGBL_CPU_CSWITCH_TIME_CUM

GBL_CPU_CSWITCH_UTILGBL_CPU_CSWITCH_UTIL

GBL_CPU_CSWITCH_UTIL_CUMGBL_CPU_CSWITCH_UTIL_CUM

GBL_CPU_CSWITCH_UTIL_HIGHGBL_CPU_CSWITCH_UTIL_HIGH

GBL_CPU_IDLE_TIMEGBL_CPU_IDLE_TIME

GBL_CPU_IDLE_TIME_CUMGBL_CPU_IDLE_TIME_CUM

GBL_CPU_IDLE_UTILGBL_CPU_IDLE_UTIL

GBL_CPU_IDLE_UTIL_CUMGBL_CPU_IDLE_UTIL_CUM

GBL_CPU_IDLE_UTIL_HIGHGBL_CPU_IDLE_UTIL_HIGH

GBL_CPU_INTERRUPT_TIMEGBL_CPU_INTERRUPT_TIME

GBL_CPU_INTERRUPT_TIME_CUMGBL_CPU_INTERRUPT_TIME_CUM

GBL_CPU_INTERRUPT_UTILGBL_CPU_INTERRUPT_UTIL

GBL_CPU_INTERRUPT_UTIL_CUMGBL_CPU_INTERRUPT_UTIL_CUM

GBL_CPU_INTERRUPT_UTIL_HIGHGBL_CPU_INTERRUPT_UTIL_HIGH

GBL_CPU_MT_ENABLEDGBL_CPU_MT_ENABLED

GBL_CPU_NICE_TIMEGBL_CPU_NICE_TIME

GBL_CPU_NICE_TIME_CUMGBL_CPU_NICE_TIME_CUM

GBL_CPU_NICE_UTILGBL_CPU_NICE_UTIL

GBL_CPU_NICE_UTIL_CUMGBL_CPU_NICE_UTIL_CUM

GBL_CPU_NICE_UTIL_HIGHGBL_CPU_NICE_UTIL_HIGH

GBL_CPU_NNICE_TIMEGBL_CPU_NNICE_TIME

GBL_CPU_NNICE_TIME_CUMGBL_CPU_NNICE_TIME_CUM

GBL_CPU_NNICE_UTILGBL_CPU_NNICE_UTIL

GBL_CPU_NNICE_UTIL_CUMGBL_CPU_NNICE_UTIL_CUM

GBL_CPU_NNICE_UTIL_HIGHGBL_CPU_NNICE_UTIL_HIGH

GBL_CPU_NORMAL_TIMEGBL_CPU_NORMAL_TIME

GBL_CPU_NORMAL_TIME_CUMGBL_CPU_NORMAL_TIME_CUM

GBL_CPU_NORMAL_UTILGBL_CPU_NORMAL_UTIL

GBL_CPU_NORMAL_UTIL_CUMGBL_CPU_NORMAL_UTIL_CUM

GBL_CPU_NORMAL_UTIL_HIGHGBL_CPU_NORMAL_UTIL_HIGH

GBL_CPU_QUEUEGBL_CPU_QUEUE

GBL_CPU_REALTIME_TIMEGBL_CPU_REALTIME_TIME

GBL_CPU_REALTIME_TIME_CUMGBL_CPU_REALTIME_TIME_CUM

GBL_CPU_REALTIME_UTILGBL_CPU_REALTIME_UTIL

GBL_CPU_REALTIME_UTIL_CUMGBL_CPU_REALTIME_UTIL_CUM

GBL_CPU_REALTIME_UTIL_HIGHGBL_CPU_REALTIME_UTIL_HIGH

GBL_CPU_SYSCALL_TIMEGBL_CPU_SYSCALL_TIME

GBL_CPU_SYSCALL_TIME_CUMGBL_CPU_SYSCALL_TIME_CUM

GBL_CPU_SYSCALL_UTILGBL_CPU_SYSCALL_UTIL

GBL_CPU_SYSCALL_UTIL_CUMGBL_CPU_SYSCALL_UTIL_CUM

GBL_CPU_SYSCALL_UTIL_HIGHGBL_CPU_SYSCALL_UTIL_HIGH

GBL_CPU_SYS_MODE_TIMEGBL_CPU_SYS_MODE_TIME

GBL_CPU_SYS_MODE_TIME_CUMGBL_CPU_SYS_MODE_TIME_CUM

GBL_CPU_SYS_MODE_UTILGBL_CPU_SYS_MODE_UTIL

GBL_CPU_SYS_MODE_UTIL_CUMGBL_CPU_SYS_MODE_UTIL_CUM

GBL_CPU_TOTAL_TIMEGBL_CPU_TOTAL_TIME

GBL_CPU_TOTAL_TIME_CUMGBL_CPU_TOTAL_TIME_CUM

GBL_CPU_TOTAL_UTILGBL_CPU_TOTAL_UTIL

GBL_CPU_TOTAL_UTIL_CUMGBL_CPU_TOTAL_UTIL_CUM

GBL_CPU_TOTAL_UTIL_HIGHGBL_CPU_TOTAL_UTIL_HIGH

GBL_CPU_TRAP_TIMEGBL_CPU_TRAP_TIME

GBL_CPU_TRAP_TIME_CUMGBL_CPU_TRAP_TIME_CUM

GBL_CPU_TRAP_UTILGBL_CPU_TRAP_UTIL

GBL_CPU_TRAP_UTIL_CUMGBL_CPU_TRAP_UTIL_CUM

GBL_CPU_TRAP_UTIL_HIGHGBL_CPU_TRAP_UTIL_HIGH

GBL_CPU_USER_MODE_TIMEGBL_CPU_USER_MODE_TIME

GBL_CPU_USER_MODE_TIME_CUMGBL_CPU_USER_MODE_TIME_CUM

GBL_CPU_USER_MODE_UTILGBL_CPU_USER_MODE_UTIL

GBL_CPU_USER_MODE_UTIL_CUMGBL_CPU_USER_MODE_UTIL_CUM

GBL_CPU_VFAULT_TIMEGBL_CPU_VFAULT_TIME

GBL_CPU_VFAULT_TIME_CUMGBL_CPU_VFAULT_TIME_CUM

GBL_CPU_VFAULT_UTILGBL_CPU_VFAULT_UTIL

GBL_CPU_VFAULT_UTIL_CUMGBL_CPU_VFAULT_UTIL_CUM

GBL_CPU_VFAULT_UTIL_HIGHGBL_CPU_VFAULT_UTIL_HIGH

GBL_CPU_WAIT_UTILGBL_CPU_WAIT_UTIL

GBL_CSWITCH_RATEGBL_CSWITCH_RATE

GBL_CSWITCH_RATE_CUMGBL_CSWITCH_RATE_CUM

GBL_CSWITCH_RATE_HIGHGBL_CSWITCH_RATE_HIGH

GBL_DISK_FS_BYTEGBL_DISK_FS_BYTE

GBL_DISK_FS_BYTE_CUMGBL_DISK_FS_BYTE_CUM

GBL_DISK_FS_IOGBL_DISK_FS_IO

GBL_DISK_FS_IO_CUMGBL_DISK_FS_IO_CUM

GBL_DISK_FS_IO_PCTGBL_DISK_FS_IO_PCT

GBL_DISK_FS_IO_PCT_CUMGBL_DISK_FS_IO_PCT_CUM

GBL_DISK_FS_IO_RATEGBL_DISK_FS_IO_RATE

GBL_DISK_FS_IO_RATE_CUMGBL_DISK_FS_IO_RATE_CUM

GBL_DISK_FS_READGBL_DISK_FS_READ

GBL_DISK_FS_READ_RATEGBL_DISK_FS_READ_RATE

GBL_DISK_FS_WRITEGBL_DISK_FS_WRITE

GBL_DISK_FS_WRITE_RATEGBL_DISK_FS_WRITE_RATE

GBL_DISK_LOGL_BYTE_RATEGBL_DISK_LOGL_BYTE_RATE

GBL_DISK_LOGL_IOGBL_DISK_LOGL_IO

GBL_DISK_LOGL_IO_CUMGBL_DISK_LOGL_IO_CUM

GBL_DISK_LOGL_IO_RATEGBL_DISK_LOGL_IO_RATE

GBL_DISK_LOGL_IO_RATE_CUMGBL_DISK_LOGL_IO_RATE_CUM

GBL_DISK_LOGL_READGBL_DISK_LOGL_READ

GBL_DISK_LOGL_READ_BYTEGBL_DISK_LOGL_READ_BYTE

GBL_DISK_LOGL_READ_BYTE_CUMGBL_DISK_LOGL_READ_BYTE_CUM

GBL_DISK_LOGL_READ_BYTE_RATEGBL_DISK_LOGL_READ_BYTE_RATE

GBL_DISK_LOGL_READ_CUMGBL_DISK_LOGL_READ_CUM

GBL_DISK_LOGL_READ_PCTGBL_DISK_LOGL_READ_PCT

GBL_DISK_LOGL_READ_PCT_CUMGBL_DISK_LOGL_READ_PCT_CUM

GBL_DISK_LOGL_READ_RATEGBL_DISK_LOGL_READ_RATE

GBL_DISK_LOGL_READ_RATE_CUMGBL_DISK_LOGL_READ_RATE_CUM

GBL_DISK_LOGL_WRITEGBL_DISK_LOGL_WRITE

GBL_DISK_LOGL_WRITE_BYTEGBL_DISK_LOGL_WRITE_BYTE

GBL_DISK_LOGL_WRITE_BYTE_CUMGBL_DISK_LOGL_WRITE_BYTE_CUM

GBL_DISK_LOGL_WRITE_BYTE_RATEGBL_DISK_LOGL_WRITE_BYTE_RATE

GBL_DISK_LOGL_WRITE_CUMGBL_DISK_LOGL_WRITE_CUM

GBL_DISK_LOGL_WRITE_PCTGBL_DISK_LOGL_WRITE_PCT

GBL_DISK_LOGL_WRITE_PCT_CUMGBL_DISK_LOGL_WRITE_PCT_CUM

GBL_DISK_LOGL_WRITE_RATEGBL_DISK_LOGL_WRITE_RATE

GBL_DISK_LOGL_WRITE_RATE_CUMGBL_DISK_LOGL_WRITE_RATE_CUM

GBL_DISK_PHYS_BYTEGBL_DISK_PHYS_BYTE

GBL_DISK_PHYS_BYTE_RATEGBL_DISK_PHYS_BYTE_RATE

GBL_DISK_PHYS_IOGBL_DISK_PHYS_IO

GBL_DISK_PHYS_IO_CUMGBL_DISK_PHYS_IO_CUM

GBL_DISK_PHYS_IO_RATEGBL_DISK_PHYS_IO_RATE

GBL_DISK_PHYS_IO_RATE_CUMGBL_DISK_PHYS_IO_RATE_CUM

GBL_DISK_PHYS_READGBL_DISK_PHYS_READ

GBL_DISK_PHYS_READ_BYTEGBL_DISK_PHYS_READ_BYTE

GBL_DISK_PHYS_READ_BYTE_CUMGBL_DISK_PHYS_READ_BYTE_CUM

GBL_DISK_PHYS_READ_BYTE_RATEGBL_DISK_PHYS_READ_BYTE_RATE

GBL_DISK_PHYS_READ_CUMGBL_DISK_PHYS_READ_CUM

GBL_DISK_PHYS_READ_PCTGBL_DISK_PHYS_READ_PCT

GBL_DISK_PHYS_READ_PCT_CUMGBL_DISK_PHYS_READ_PCT_CUM

GBL_DISK_PHYS_READ_RATEGBL_DISK_PHYS_READ_RATE

GBL_DISK_PHYS_READ_RATE_CUMGBL_DISK_PHYS_READ_RATE_CUM

GBL_DISK_PHYS_WRITEGBL_DISK_PHYS_WRITE

GBL_DISK_PHYS_WRITE_BYTEGBL_DISK_PHYS_WRITE_BYTE

GBL_DISK_PHYS_WRITE_BYTE_CUMGBL_DISK_PHYS_WRITE_BYTE_CUM

GBL_DISK_PHYS_WRITE_BYTE_RATEGBL_DISK_PHYS_WRITE_BYTE_RATE

GBL_DISK_PHYS_WRITE_CUMGBL_DISK_PHYS_WRITE_CUM

GBL_DISK_PHYS_WRITE_PCTGBL_DISK_PHYS_WRITE_PCT

GBL_DISK_PHYS_WRITE_PCT_CUMGBL_DISK_PHYS_WRITE_PCT_CUM

GBL_DISK_PHYS_WRITE_RATEGBL_DISK_PHYS_WRITE_RATE

GBL_DISK_PHYS_WRITE_RATE_CUMGBL_DISK_PHYS_WRITE_RATE_CUM

GBL_DISK_QUEUEGBL_DISK_QUEUE

GBL_DISK_RAW_BYTEGBL_DISK_RAW_BYTE

GBL_DISK_RAW_BYTE_CUMGBL_DISK_RAW_BYTE_CUM

GBL_DISK_RAW_IOGBL_DISK_RAW_IO

GBL_DISK_RAW_IO_CUMGBL_DISK_RAW_IO_CUM

GBL_DISK_RAW_IO_PCTGBL_DISK_RAW_IO_PCT

GBL_DISK_RAW_IO_PCT_CUMGBL_DISK_RAW_IO_PCT_CUM

GBL_DISK_RAW_IO_RATEGBL_DISK_RAW_IO_RATE

GBL_DISK_RAW_IO_RATE_CUMGBL_DISK_RAW_IO_RATE_CUM

GBL_DISK_RAW_READGBL_DISK_RAW_READ

GBL_DISK_RAW_READ_RATEGBL_DISK_RAW_READ_RATE

GBL_DISK_RAW_WRITEGBL_DISK_RAW_WRITE

GBL_DISK_RAW_WRITE_RATEGBL_DISK_RAW_WRITE_RATE

GBL_DISK_REM_FS_BYTEGBL_DISK_REM_FS_BYTE

GBL_DISK_REM_FS_BYTE_CUMGBL_DISK_REM_FS_BYTE_CUM

GBL_DISK_REM_FS_IOGBL_DISK_REM_FS_IO

GBL_DISK_REM_FS_IO_CUMGBL_DISK_REM_FS_IO_CUM

GBL_DISK_REM_FS_IO_PCTGBL_DISK_REM_FS_IO_PCT

GBL_DISK_REM_FS_IO_PCT_CUMGBL_DISK_REM_FS_IO_PCT_CUM

GBL_DISK_REM_FS_IO_RATEGBL_DISK_REM_FS_IO_RATE

GBL_DISK_REM_FS_IO_RATE_CUMGBL_DISK_REM_FS_IO_RATE_CUM

GBL_DISK_REM_LOGL_READGBL_DISK_REM_LOGL_READ

GBL_DISK_REM_LOGL_READ_BYTEGBL_DISK_REM_LOGL_READ_BYTE

GBL_DISK_REM_LOGL_READ_BYTE_CUMGBL_DISK_REM_LOGL_READ_BYTE_CUM

GBL_DISK_REM_LOGL_READ_CUMGBL_DISK_REM_LOGL_READ_CUM

GBL_DISK_REM_LOGL_READ_PCTGBL_DISK_REM_LOGL_READ_PCT

GBL_DISK_REM_LOGL_READ_PCT_CUMGBL_DISK_REM_LOGL_READ_PCT_CUM

GBL_DISK_REM_LOGL_READ_RATEGBL_DISK_REM_LOGL_READ_RATE

GBL_DISK_REM_LOGL_READ_RATE_CUMGBL_DISK_REM_LOGL_READ_RATE_CUM

GBL_DISK_REM_LOGL_WRITEGBL_DISK_REM_LOGL_WRITE

GBL_DISK_REM_LOGL_WRITE_BYTEGBL_DISK_REM_LOGL_WRITE_BYTE

GBL_DISK_REM_LOGL_WRITE_BYTE_CUMGBL_DISK_REM_LOGL_WRITE_BYTE_CUM

GBL_DISK_REM_LOGL_WRITE_CUMGBL_DISK_REM_LOGL_WRITE_CUM

GBL_DISK_REM_LOGL_WRITE_PCTGBL_DISK_REM_LOGL_WRITE_PCT

GBL_DISK_REM_LOGL_WRITE_PCT_CUMGBL_DISK_REM_LOGL_WRITE_PCT_CUM

GBL_DISK_REM_LOGL_WRITE_RATEGBL_DISK_REM_LOGL_WRITE_RATE

GBL_DISK_REM_LOGL_WRITE_RATE_CUMGBL_DISK_REM_LOGL_WRITE_RATE_CUM

GBL_DISK_REM_PHYS_READGBL_DISK_REM_PHYS_READ

GBL_DISK_REM_PHYS_READ_BYTEGBL_DISK_REM_PHYS_READ_BYTE

GBL_DISK_REM_PHYS_READ_BYTE_CUMGBL_DISK_REM_PHYS_READ_BYTE_CUM

GBL_DISK_REM_PHYS_READ_CUMGBL_DISK_REM_PHYS_READ_CUM

GBL_DISK_REM_PHYS_READ_PCTGBL_DISK_REM_PHYS_READ_PCT

GBL_DISK_REM_PHYS_READ_PCT_CUMGBL_DISK_REM_PHYS_READ_PCT_CUM

GBL_DISK_REM_PHYS_READ_RATEGBL_DISK_REM_PHYS_READ_RATE

GBL_DISK_REM_PHYS_READ_RATE_CUMGBL_DISK_REM_PHYS_READ_RATE_CUM

GBL_DISK_REM_PHYS_WRITEGBL_DISK_REM_PHYS_WRITE

GBL_DISK_REM_PHYS_WRITE_BYTEGBL_DISK_REM_PHYS_WRITE_BYTE

GBL_DISK_REM_PHYS_WRITE_BYTE_CUMGBL_DISK_REM_PHYS_WRITE_BYTE_CUM

GBL_DISK_REM_PHYS_WRITE_CUMGBL_DISK_REM_PHYS_WRITE_CUM

GBL_DISK_REM_PHYS_WRITE_PCTGBL_DISK_REM_PHYS_WRITE_PCT

GBL_DISK_REM_PHYS_WRITE_PCT_CUMGBL_DISK_REM_PHYS_WRITE_PCT_CUM

GBL_DISK_REM_PHYS_WRITE_RATEGBL_DISK_REM_PHYS_WRITE_RATE

GBL_DISK_REM_PHYS_WRITE_RATE_CUMGBL_DISK_REM_PHYS_WRITE_RATE_CUM

GBL_DISK_REM_RAW_BYTEGBL_DISK_REM_RAW_BYTE

GBL_DISK_REM_RAW_BYTE_CUMGBL_DISK_REM_RAW_BYTE_CUM

GBL_DISK_REM_RAW_IOGBL_DISK_REM_RAW_IO

GBL_DISK_REM_RAW_IO_CUMGBL_DISK_REM_RAW_IO_CUM

GBL_DISK_REM_RAW_IO_PCTGBL_DISK_REM_RAW_IO_PCT

GBL_DISK_REM_RAW_IO_PCT_CUMGBL_DISK_REM_RAW_IO_PCT_CUM

GBL_DISK_REM_RAW_IO_RATEGBL_DISK_REM_RAW_IO_RATE

GBL_DISK_REM_RAW_IO_RATE_CUMGBL_DISK_REM_RAW_IO_RATE_CUM

GBL_DISK_REM_SYSTEM_BYTEGBL_DISK_REM_SYSTEM_BYTE

GBL_DISK_REM_SYSTEM_BYTE_CUMGBL_DISK_REM_SYSTEM_BYTE_CUM

GBL_DISK_REM_SYSTEM_IOGBL_DISK_REM_SYSTEM_IO

GBL_DISK_REM_SYSTEM_IO_CUMGBL_DISK_REM_SYSTEM_IO_CUM

GBL_DISK_REM_SYSTEM_IO_PCTGBL_DISK_REM_SYSTEM_IO_PCT

GBL_DISK_REM_SYSTEM_IO_PCT_CUMGBL_DISK_REM_SYSTEM_IO_PCT_CUM

GBL_DISK_REM_SYSTEM_IO_RATEGBL_DISK_REM_SYSTEM_IO_RATE

GBL_DISK_REM_SYSTEM_IO_RATE_CUMGBL_DISK_REM_SYSTEM_IO_RATE_CUM

GBL_DISK_REM_VM_BYTEGBL_DISK_REM_VM_BYTE

GBL_DISK_REM_VM_BYTE_CUMGBL_DISK_REM_VM_BYTE_CUM

GBL_DISK_REM_VM_IOGBL_DISK_REM_VM_IO

GBL_DISK_REM_VM_IO_CUMGBL_DISK_REM_VM_IO_CUM

GBL_DISK_REM_VM_IO_PCTGBL_DISK_REM_VM_IO_PCT

GBL_DISK_REM_VM_IO_PCT_CUMGBL_DISK_REM_VM_IO_PCT_CUM

GBL_DISK_REM_VM_IO_RATEGBL_DISK_REM_VM_IO_RATE

GBL_DISK_REM_VM_IO_RATE_CUMGBL_DISK_REM_VM_IO_RATE_CUM

GBL_DISK_REQUEST_QUEUEGBL_DISK_REQUEST_QUEUE

GBL_DISK_SUBSYSTEM_QUEUEGBL_DISK_SUBSYSTEM_QUEUE

GBL_DISK_SUBSYSTEM_WAIT_PCTGBL_DISK_SUBSYSTEM_WAIT_PCT

GBL_DISK_SYSTEM_BYTEGBL_DISK_SYSTEM_BYTE

GBL_DISK_SYSTEM_BYTE_CUMGBL_DISK_SYSTEM_BYTE_CUM

GBL_DISK_SYSTEM_IOGBL_DISK_SYSTEM_IO

GBL_DISK_SYSTEM_IO_CUMGBL_DISK_SYSTEM_IO_CUM

GBL_DISK_SYSTEM_IO_PCTGBL_DISK_SYSTEM_IO_PCT

GBL_DISK_SYSTEM_IO_PCT_CUMGBL_DISK_SYSTEM_IO_PCT_CUM

GBL_DISK_SYSTEM_IO_RATEGBL_DISK_SYSTEM_IO_RATE

GBL_DISK_SYSTEM_IO_RATE_CUMGBL_DISK_SYSTEM_IO_RATE_CUM

GBL_DISK_SYSTEM_READGBL_DISK_SYSTEM_READ

GBL_DISK_SYSTEM_READ_RATEGBL_DISK_SYSTEM_READ_RATE

GBL_DISK_SYSTEM_WRITEGBL_DISK_SYSTEM_WRITE

GBL_DISK_SYSTEM_WRITE_RATEGBL_DISK_SYSTEM_WRITE_RATE

GBL_DISK_TIME_PEAKGBL_DISK_TIME_PEAK

GBL_DISK_UTILGBL_DISK_UTIL

GBL_DISK_UTIL_PEAKGBL_DISK_UTIL_PEAK

GBL_DISK_UTIL_PEAK_CUMGBL_DISK_UTIL_PEAK_CUM

GBL_DISK_UTIL_PEAK_HIGHGBL_DISK_UTIL_PEAK_HIGH

GBL_DISK_UTIL_PEAK_OTHERSGBL_DISK_UTIL_PEAK_OTHERS

GBL_DISK_UTIL_PEAK_VMGBL_DISK_UTIL_PEAK_VM

GBL_DISK_VM_BYTEGBL_DISK_VM_BYTE

GBL_DISK_VM_BYTE_CUMGBL_DISK_VM_BYTE_CUM

GBL_DISK_VM_IOGBL_DISK_VM_IO

GBL_DISK_VM_IO_CUMGBL_DISK_VM_IO_CUM

GBL_DISK_VM_IO_PCTGBL_DISK_VM_IO_PCT

GBL_DISK_VM_IO_PCT_CUMGBL_DISK_VM_IO_PCT_CUM

GBL_DISK_VM_IO_RATEGBL_DISK_VM_IO_RATE

GBL_DISK_VM_IO_RATE_CUMGBL_DISK_VM_IO_RATE_CUM

GBL_DISK_VM_READGBL_DISK_VM_READ

GBL_DISK_VM_READ_CUMGBL_DISK_VM_READ_CUM

GBL_DISK_VM_READ_RATEGBL_DISK_VM_READ_RATE

GBL_DISK_VM_READ_RATE_CUMGBL_DISK_VM_READ_RATE_CUM

GBL_DISK_VM_READ_RATE_HIGHGBL_DISK_VM_READ_RATE_HIGH

GBL_DISK_VM_WRITEGBL_DISK_VM_WRITE

GBL_DISK_VM_WRITE_CUMGBL_DISK_VM_WRITE_CUM

GBL_DISK_VM_WRITE_RATEGBL_DISK_VM_WRITE_RATE

GBL_DISK_VM_WRITE_RATE_CUMGBL_DISK_VM_WRITE_RATE_CUM

GBL_DISK_VM_WRITE_RATE_HIGHGBL_DISK_VM_WRITE_RATE_HIGH

GBL_DISK_WAIT_PCTGBL_DISK_WAIT_PCT

GBL_DISK_WAIT_TIMEGBL_DISK_WAIT_TIME

GBL_FS_SPACE_UTIL_PEAKGBL_FS_SPACE_UTIL_PEAK

GBL_GMTOFFSETGBL_GMTOFFSET

GBL_GRAPHICS_QUEUEGBL_GRAPHICS_QUEUE

GBL_GRAPHICS_WAIT_PCTGBL_GRAPHICS_WAIT_PCT

GBL_GRAPHICS_WAIT_TIMEGBL_GRAPHICS_WAIT_TIME

GBL_IGNORE_MTGBL_IGNORE_MT

GBL_INODE_QUEUEGBL_INODE_QUEUE

GBL_INODE_WAIT_PCTGBL_INODE_WAIT_PCT

GBL_INODE_WAIT_TIMEGBL_INODE_WAIT_TIME

GBL_INTERRUPTGBL_INTERRUPT

GBL_INTERRUPT_RATEGBL_INTERRUPT_RATE

GBL_INTERRUPT_RATE_CUMGBL_INTERRUPT_RATE_CUM

GBL_INTERRUPT_RATE_HIGHGBL_INTERRUPT_RATE_HIGH

GBL_INTERVALGBL_INTERVAL

GBL_INTERVAL_CUMGBL_INTERVAL_CUM

GBL_IPC_QUEUEGBL_IPC_QUEUE

GBL_IPC_SUBSYSTEM_QUEUEGBL_IPC_SUBSYSTEM_QUEUE

GBL_IPC_SUBSYSTEM_WAIT_PCTGBL_IPC_SUBSYSTEM_WAIT_PCT

GBL_IPC_WAIT_PCTGBL_IPC_WAIT_PCT

GBL_IPC_WAIT_TIMEGBL_IPC_WAIT_TIME

GBL_JAVAARGGBL_JAVAARG

GBL_JOBCTL_QUEUEGBL_JOBCTL_QUEUE

GBL_JOBCTL_WAIT_PCTGBL_JOBCTL_WAIT_PCT

GBL_JOBCTL_WAIT_TIMEGBL_JOBCTL_WAIT_TIME

GBL_LAN_QUEUEGBL_LAN_QUEUE

GBL_LAN_WAIT_PCTGBL_LAN_WAIT_PCT

GBL_LAN_WAIT_TIMEGBL_LAN_WAIT_TIME

GBL_LOADAVGGBL_LOADAVG

GBL_LOADAVG15GBL_LOADAVG15

GBL_LOADAVG5GBL_LOADAVG5

GBL_LOST_MI_TRACE_BUFFERSGBL_LOST_MI_TRACE_BUFFERS

GBL_LS_ROLEGBL_LS_ROLE

GBL_LS_TYPEGBL_LS_TYPE

GBL_LS_UUIDGBL_LS_UUID

GBL_MACHINEGBL_MACHINE

GBL_MACHINE_MODELGBL_MACHINE_MODEL

GBL_MEMFS_BLK_CNTGBL_MEMFS_BLK_CNT

GBL_MEMFS_SWP_CNTGBL_MEMFS_SWP_CNT

GBL_MEM_ACTIVE_VIRTGBL_MEM_ACTIVE_VIRT

GBL_MEM_ACTIVE_VIRT_UTILGBL_MEM_ACTIVE_VIRT_UTIL

GBL_MEM_AVAILGBL_MEM_AVAIL

GBL_MEM_CACHEGBL_MEM_CACHE

GBL_MEM_CACHE_HITGBL_MEM_CACHE_HIT

GBL_MEM_CACHE_HIT_CUMGBL_MEM_CACHE_HIT_CUM

GBL_MEM_CACHE_HIT_PCTGBL_MEM_CACHE_HIT_PCT

GBL_MEM_CACHE_HIT_PCT_CUMGBL_MEM_CACHE_HIT_PCT_CUM

GBL_MEM_CACHE_HIT_PCT_HIGHGBL_MEM_CACHE_HIT_PCT_HIGH

GBL_MEM_CACHE_UTILGBL_MEM_CACHE_UTIL

GBL_MEM_CACHE_WRITE_HITGBL_MEM_CACHE_WRITE_HIT

GBL_MEM_CACHE_WRITE_HIT_CUMGBL_MEM_CACHE_WRITE_HIT_CUM

GBL_MEM_CACHE_WRITE_HIT_PCTGBL_MEM_CACHE_WRITE_HIT_PCT

GBL_MEM_CACHE_WRITE_HIT_PCT_CUMGBL_MEM_CACHE_WRITE_HIT_PCT_CUM

GBL_MEM_DNLC_HITGBL_MEM_DNLC_HIT

GBL_MEM_DNLC_HIT_CUMGBL_MEM_DNLC_HIT_CUM

GBL_MEM_DNLC_HIT_PCTGBL_MEM_DNLC_HIT_PCT

GBL_MEM_DNLC_HIT_PCT_CUMGBL_MEM_DNLC_HIT_PCT_CUM

GBL_MEM_DNLC_HIT_PCT_HIGHGBL_MEM_DNLC_HIT_PCT_HIGH

GBL_MEM_DNLC_LONGSGBL_MEM_DNLC_LONGS

GBL_MEM_DNLC_LONGS_CUMGBL_MEM_DNLC_LONGS_CUM

GBL_MEM_DNLC_LONGS_PCTGBL_MEM_DNLC_LONGS_PCT

GBL_MEM_DNLC_LONGS_PCT_CUMGBL_MEM_DNLC_LONGS_PCT_CUM

GBL_MEM_DNLC_LONGS_PCT_HIGHGBL_MEM_DNLC_LONGS_PCT_HIGH

GBL_MEM_FILE_PAGE_CACHEGBL_MEM_FILE_PAGE_CACHE

GBL_MEM_FILE_PAGE_CACHE_UTILGBL_MEM_FILE_PAGE_CACHE_UTIL

GBL_MEM_FREEGBL_MEM_FREE

GBL_MEM_FREE_UTILGBL_MEM_FREE_UTIL

GBL_MEM_PAGEINGBL_MEM_PAGEIN

GBL_MEM_PAGEIN_BYTEGBL_MEM_PAGEIN_BYTE

GBL_MEM_PAGEIN_BYTE_CUMGBL_MEM_PAGEIN_BYTE_CUM

GBL_MEM_PAGEIN_BYTE_RATEGBL_MEM_PAGEIN_BYTE_RATE

GBL_MEM_PAGEIN_BYTE_RATE_CUMGBL_MEM_PAGEIN_BYTE_RATE_CUM

GBL_MEM_PAGEIN_BYTE_RATE_HIGHGBL_MEM_PAGEIN_BYTE_RATE_HIGH

GBL_MEM_PAGEIN_CUMGBL_MEM_PAGEIN_CUM

GBL_MEM_PAGEIN_RATEGBL_MEM_PAGEIN_RATE

GBL_MEM_PAGEIN_RATE_CUMGBL_MEM_PAGEIN_RATE_CUM

GBL_MEM_PAGEIN_RATE_HIGHGBL_MEM_PAGEIN_RATE_HIGH

GBL_MEM_PAGEOUTGBL_MEM_PAGEOUT

GBL_MEM_PAGEOUT_BYTEGBL_MEM_PAGEOUT_BYTE

GBL_MEM_PAGEOUT_BYTE_CUMGBL_MEM_PAGEOUT_BYTE_CUM

GBL_MEM_PAGEOUT_BYTE_RATEGBL_MEM_PAGEOUT_BYTE_RATE

GBL_MEM_PAGEOUT_BYTE_RATE_CUMGBL_MEM_PAGEOUT_BYTE_RATE_CUM

GBL_MEM_PAGEOUT_BYTE_RATE_HIGHGBL_MEM_PAGEOUT_BYTE_RATE_HIGH

GBL_MEM_PAGEOUT_CUMGBL_MEM_PAGEOUT_CUM

GBL_MEM_PAGEOUT_RATEGBL_MEM_PAGEOUT_RATE

GBL_MEM_PAGEOUT_RATE_CUMGBL_MEM_PAGEOUT_RATE_CUM

GBL_MEM_PAGEOUT_RATE_HIGHGBL_MEM_PAGEOUT_RATE_HIGH

GBL_MEM_PAGE_FAULTGBL_MEM_PAGE_FAULT

GBL_MEM_PAGE_FAULT_CUMGBL_MEM_PAGE_FAULT_CUM

GBL_MEM_PAGE_FAULT_RATEGBL_MEM_PAGE_FAULT_RATE

GBL_MEM_PAGE_FAULT_RATE_CUMGBL_MEM_PAGE_FAULT_RATE_CUM

GBL_MEM_PAGE_FAULT_RATE_HIGHGBL_MEM_PAGE_FAULT_RATE_HIGH

GBL_MEM_PAGE_REQUESTGBL_MEM_PAGE_REQUEST

GBL_MEM_PAGE_REQUEST_CUMGBL_MEM_PAGE_REQUEST_CUM

GBL_MEM_PAGE_REQUEST_RATEGBL_MEM_PAGE_REQUEST_RATE

GBL_MEM_PAGE_REQUEST_RATE_CUMGBL_MEM_PAGE_REQUEST_RATE_CUM

GBL_MEM_PAGE_REQUEST_RATE_HIGHGBL_MEM_PAGE_REQUEST_RATE_HIGH

GBL_MEM_PAGE_SIZE_MAXGBL_MEM_PAGE_SIZE_MAX

GBL_MEM_PG_SCANGBL_MEM_PG_SCAN

GBL_MEM_PG_SCAN_CUMGBL_MEM_PG_SCAN_CUM

GBL_MEM_PG_SCAN_RATEGBL_MEM_PG_SCAN_RATE

GBL_MEM_PG_SCAN_RATE_CUMGBL_MEM_PG_SCAN_RATE_CUM

GBL_MEM_PG_SCAN_RATE_HIGHGBL_MEM_PG_SCAN_RATE_HIGH

GBL_MEM_PHYSGBL_MEM_PHYS

GBL_MEM_QUEUEGBL_MEM_QUEUE

GBL_MEM_SWAPGBL_MEM_SWAP

GBL_MEM_SWAPINGBL_MEM_SWAPIN

GBL_MEM_SWAPIN_BYTEGBL_MEM_SWAPIN_BYTE

GBL_MEM_SWAPIN_BYTE_CUMGBL_MEM_SWAPIN_BYTE_CUM

GBL_MEM_SWAPIN_BYTE_RATEGBL_MEM_SWAPIN_BYTE_RATE

GBL_MEM_SWAPIN_BYTE_RATE_CUMGBL_MEM_SWAPIN_BYTE_RATE_CUM

GBL_MEM_SWAPIN_BYTE_RATE_HIGHGBL_MEM_SWAPIN_BYTE_RATE_HIGH

GBL_MEM_SWAPIN_CUMGBL_MEM_SWAPIN_CUM

GBL_MEM_SWAPIN_RATEGBL_MEM_SWAPIN_RATE

GBL_MEM_SWAPIN_RATE_CUMGBL_MEM_SWAPIN_RATE_CUM

GBL_MEM_SWAPIN_RATE_HIGHGBL_MEM_SWAPIN_RATE_HIGH

GBL_MEM_SWAPOUTGBL_MEM_SWAPOUT

GBL_MEM_SWAPOUT_BYTEGBL_MEM_SWAPOUT_BYTE

GBL_MEM_SWAPOUT_BYTE_CUMGBL_MEM_SWAPOUT_BYTE_CUM

GBL_MEM_SWAPOUT_BYTE_RATEGBL_MEM_SWAPOUT_BYTE_RATE

GBL_MEM_SWAPOUT_BYTE_RATE_CUMGBL_MEM_SWAPOUT_BYTE_RATE_CUM

GBL_MEM_SWAPOUT_BYTE_RATE_HIGHGBL_MEM_SWAPOUT_BYTE_RATE_HIGH

GBL_MEM_SWAPOUT_CUMGBL_MEM_SWAPOUT_CUM

GBL_MEM_SWAPOUT_RATEGBL_MEM_SWAPOUT_RATE

GBL_MEM_SWAPOUT_RATE_CUMGBL_MEM_SWAPOUT_RATE_CUM

GBL_MEM_SWAPOUT_RATE_HIGHGBL_MEM_SWAPOUT_RATE_HIGH

GBL_MEM_SWAP_1_MIN_RATEGBL_MEM_SWAP_1_MIN_RATE

GBL_MEM_SWAP_CUMGBL_MEM_SWAP_CUM

GBL_MEM_SWAP_RATEGBL_MEM_SWAP_RATE

GBL_MEM_SWAP_RATE_CUMGBL_MEM_SWAP_RATE_CUM

GBL_MEM_SWAP_RATE_HIGHGBL_MEM_SWAP_RATE_HIGH

GBL_MEM_SYSGBL_MEM_SYS

GBL_MEM_SYS_AND_CACHE_UTILGBL_MEM_SYS_AND_CACHE_UTIL

GBL_MEM_SYS_UTILGBL_MEM_SYS_UTIL

GBL_MEM_USERGBL_MEM_USER

GBL_MEM_USER_UTILGBL_MEM_USER_UTIL

GBL_MEM_UTILGBL_MEM_UTIL

GBL_MEM_UTIL_CUMGBL_MEM_UTIL_CUM

GBL_MEM_UTIL_HIGHGBL_MEM_UTIL_HIGH

GBL_MEM_VIRTGBL_MEM_VIRT

GBL_MEM_WAIT_PCTGBL_MEM_WAIT_PCT

GBL_MEM_WAIT_TIMEGBL_MEM_WAIT_TIME

GBL_MI_LOST_PROCGBL_MI_LOST_PROC

GBL_MI_LOST_PROC_CUMGBL_MI_LOST_PROC_CUM

GBL_MI_PROC_ENTRIESGBL_MI_PROC_ENTRIES

GBL_MI_THREAD_ENTRIESGBL_MI_THREAD_ENTRIES

GBL_MSG_QUEUEGBL_MSG_QUEUE

GBL_MSG_WAIT_PCTGBL_MSG_WAIT_PCT

GBL_MSG_WAIT_TIMEGBL_MSG_WAIT_TIME

GBL_NETWORK_SUBSYSTEM_QUEUEGBL_NETWORK_SUBSYSTEM_QUEUE

GBL_NETWORK_SUBSYSTEM_WAIT_PCTGBL_NETWORK_SUBSYSTEM_WAIT_PCT

GBL_NET_COLLISIONGBL_NET_COLLISION

GBL_NET_COLLISION_1_MIN_RATEGBL_NET_COLLISION_1_MIN_RATE

GBL_NET_COLLISION_CUMGBL_NET_COLLISION_CUM

GBL_NET_COLLISION_PCTGBL_NET_COLLISION_PCT

GBL_NET_COLLISION_PCT_CUMGBL_NET_COLLISION_PCT_CUM

GBL_NET_COLLISION_RATEGBL_NET_COLLISION_RATE

GBL_NET_DEFERREDGBL_NET_DEFERRED

GBL_NET_DEFERRED_CUMGBL_NET_DEFERRED_CUM

GBL_NET_DEFERRED_PCTGBL_NET_DEFERRED_PCT

GBL_NET_DEFERRED_PCT_CUMGBL_NET_DEFERRED_PCT_CUM

GBL_NET_DEFERRED_RATEGBL_NET_DEFERRED_RATE

GBL_NET_DEFERRED_RATE_CUMGBL_NET_DEFERRED_RATE_CUM

GBL_NET_ERRORGBL_NET_ERROR

GBL_NET_ERROR_1_MIN_RATEGBL_NET_ERROR_1_MIN_RATE

GBL_NET_ERROR_CUMGBL_NET_ERROR_CUM

GBL_NET_ERROR_RATEGBL_NET_ERROR_RATE

GBL_NET_IN_ERRORGBL_NET_IN_ERROR

GBL_NET_IN_ERROR_CUMGBL_NET_IN_ERROR_CUM

GBL_NET_IN_ERROR_PCTGBL_NET_IN_ERROR_PCT

GBL_NET_IN_ERROR_PCT_CUMGBL_NET_IN_ERROR_PCT_CUM

GBL_NET_IN_ERROR_RATEGBL_NET_IN_ERROR_RATE

GBL_NET_IN_ERROR_RATE_CUMGBL_NET_IN_ERROR_RATE_CUM

GBL_NET_IN_PACKETGBL_NET_IN_PACKET

GBL_NET_IN_PACKET_CUMGBL_NET_IN_PACKET_CUM

GBL_NET_IN_PACKET_RATEGBL_NET_IN_PACKET_RATE

GBL_NET_IP_FRAGMENTS_RECEIVEDGBL_NET_IP_FRAGMENTS_RECEIVED

GBL_NET_IP_FWD_DATAGRAMSGBL_NET_IP_FWD_DATAGRAMS

GBL_NET_IP_REASSEMBLY_REQUIREDGBL_NET_IP_REASSEMBLY_REQUIRED

GBL_NET_OUTQUEUEGBL_NET_OUTQUEUE

GBL_NET_OUT_ERRORGBL_NET_OUT_ERROR

GBL_NET_OUT_ERROR_CUMGBL_NET_OUT_ERROR_CUM

GBL_NET_OUT_ERROR_PCTGBL_NET_OUT_ERROR_PCT

GBL_NET_OUT_ERROR_PCT_CUMGBL_NET_OUT_ERROR_PCT_CUM

GBL_NET_OUT_ERROR_RATEGBL_NET_OUT_ERROR_RATE

GBL_NET_OUT_ERROR_RATE_CUMGBL_NET_OUT_ERROR_RATE_CUM

GBL_NET_OUT_PACKETGBL_NET_OUT_PACKET

GBL_NET_OUT_PACKET_CUMGBL_NET_OUT_PACKET_CUM

GBL_NET_OUT_PACKET_RATEGBL_NET_OUT_PACKET_RATE

GBL_NET_PACKETGBL_NET_PACKET

GBL_NET_PACKET_RATEGBL_NET_PACKET_RATE

GBL_NET_UTIL_PEAKGBL_NET_UTIL_PEAK

GBL_NFS_CALLGBL_NFS_CALL

GBL_NFS_CALL_RATEGBL_NFS_CALL_RATE

GBL_NFS_CLIENT_BAD_CALLGBL_NFS_CLIENT_BAD_CALL

GBL_NFS_CLIENT_BAD_CALL_CUMGBL_NFS_CLIENT_BAD_CALL_CUM

GBL_NFS_CLIENT_BIODGBL_NFS_CLIENT_BIOD

GBL_NFS_CLIENT_BYTEGBL_NFS_CLIENT_BYTE

GBL_NFS_CLIENT_BYTE_CUMGBL_NFS_CLIENT_BYTE_CUM

GBL_NFS_CLIENT_CALLGBL_NFS_CLIENT_CALL

GBL_NFS_CLIENT_CALL_CUMGBL_NFS_CLIENT_CALL_CUM

GBL_NFS_CLIENT_CALL_RATEGBL_NFS_CLIENT_CALL_RATE

GBL_NFS_CLIENT_IDLE_BIODGBL_NFS_CLIENT_IDLE_BIOD

GBL_NFS_CLIENT_IOGBL_NFS_CLIENT_IO

GBL_NFS_CLIENT_IO_CUMGBL_NFS_CLIENT_IO_CUM

GBL_NFS_CLIENT_IO_PCTGBL_NFS_CLIENT_IO_PCT

GBL_NFS_CLIENT_IO_PCT_CUMGBL_NFS_CLIENT_IO_PCT_CUM

GBL_NFS_CLIENT_IO_RATEGBL_NFS_CLIENT_IO_RATE

GBL_NFS_CLIENT_IO_RATE_CUMGBL_NFS_CLIENT_IO_RATE_CUM

GBL_NFS_CLIENT_PHYS_TIMEGBL_NFS_CLIENT_PHYS_TIME

GBL_NFS_CLIENT_PHYS_TIME_CUMGBL_NFS_CLIENT_PHYS_TIME_CUM

GBL_NFS_CLIENT_READ_BYTE_RATEGBL_NFS_CLIENT_READ_BYTE_RATE

GBL_NFS_CLIENT_READ_BYTE_RATE_CUMGBL_NFS_CLIENT_READ_BYTE_RATE_
CUM

GBL_NFS_CLIENT_READ_RATEGBL_NFS_CLIENT_READ_RATE

GBL_NFS_CLIENT_READ_RATE_CUMGBL_NFS_CLIENT_READ_RATE_CUM

GBL_NFS_CLIENT_SERVICE_QUEUEGBL_NFS_CLIENT_SERVICE_QUEUE

GBL_NFS_CLIENT_SERVICE_QUEUE_CUMGBL_NFS_CLIENT_SERVICE_QUEUE_CUM

GBL_NFS_CLIENT_SERVICE_TIMEGBL_NFS_CLIENT_SERVICE_TIME

GBL_NFS_CLIENT_SERVICE_TIME_CUMGBL_NFS_CLIENT_SERVICE_TIME_CUM

GBL_NFS_CLIENT_WRITE_BYTE_RATEGBL_NFS_CLIENT_WRITE_BYTE_RATE

GBL_NFS_CLIENT_WRITE_BYTE_RATE_CUMGBL_NFS_CLIENT_WRITE_BYTE_RATE_
CUM

GBL_NFS_CLIENT_WRITE_RATEGBL_NFS_CLIENT_WRITE_RATE

GBL_NFS_CLIENT_WRITE_RATE_CUMGBL_NFS_CLIENT_WRITE_RATE_CUM

GBL_NFS_LOGL_READGBL_NFS_LOGL_READ

GBL_NFS_LOGL_READ_BYTEGBL_NFS_LOGL_READ_BYTE

GBL_NFS_LOGL_READ_BYTE_CUMGBL_NFS_LOGL_READ_BYTE_CUM

GBL_NFS_LOGL_READ_CUMGBL_NFS_LOGL_READ_CUM

GBL_NFS_LOGL_READ_PCTGBL_NFS_LOGL_READ_PCT

GBL_NFS_LOGL_READ_PCT_CUMGBL_NFS_LOGL_READ_PCT_CUM

GBL_NFS_LOGL_READ_RATEGBL_NFS_LOGL_READ_RATE

GBL_NFS_LOGL_READ_RATE_CUMGBL_NFS_LOGL_READ_RATE_CUM

GBL_NFS_LOGL_WRITEGBL_NFS_LOGL_WRITE

GBL_NFS_LOGL_WRITE_BYTEGBL_NFS_LOGL_WRITE_BYTE

GBL_NFS_LOGL_WRITE_BYTE_CUMGBL_NFS_LOGL_WRITE_BYTE_CUM

GBL_NFS_LOGL_WRITE_CUMGBL_NFS_LOGL_WRITE_CUM

GBL_NFS_LOGL_WRITE_PCTGBL_NFS_LOGL_WRITE_PCT

GBL_NFS_LOGL_WRITE_PCT_CUMGBL_NFS_LOGL_WRITE_PCT_CUM

GBL_NFS_LOGL_WRITE_RATEGBL_NFS_LOGL_WRITE_RATE

GBL_NFS_LOGL_WRITE_RATE_CUMGBL_NFS_LOGL_WRITE_RATE_CUM

GBL_NFS_QUEUEGBL_NFS_QUEUE

GBL_NFS_SERVER_BAD_CALLGBL_NFS_SERVER_BAD_CALL

GBL_NFS_SERVER_BAD_CALL_CUMGBL_NFS_SERVER_BAD_CALL_CUM

GBL_NFS_SERVER_BYTEGBL_NFS_SERVER_BYTE

GBL_NFS_SERVER_BYTE_CUMGBL_NFS_SERVER_BYTE_CUM

GBL_NFS_SERVER_CALLGBL_NFS_SERVER_CALL

GBL_NFS_SERVER_CALL_CUMGBL_NFS_SERVER_CALL_CUM

GBL_NFS_SERVER_CALL_RATEGBL_NFS_SERVER_CALL_RATE

GBL_NFS_SERVER_IOGBL_NFS_SERVER_IO

GBL_NFS_SERVER_IO_CUMGBL_NFS_SERVER_IO_CUM

GBL_NFS_SERVER_IO_PCTGBL_NFS_SERVER_IO_PCT

GBL_NFS_SERVER_IO_PCT_CUMGBL_NFS_SERVER_IO_PCT_CUM

GBL_NFS_SERVER_IO_RATEGBL_NFS_SERVER_IO_RATE

GBL_NFS_SERVER_IO_RATE_CUMGBL_NFS_SERVER_IO_RATE_CUM

GBL_NFS_SERVER_READ_BYTE_RATEGBL_NFS_SERVER_READ_BYTE_RATE

GBL_NFS_SERVER_READ_BYTE_RATE_CUMGBL_NFS_SERVER_READ_BYTE_RATE_
CUM

GBL_NFS_SERVER_READ_RATEGBL_NFS_SERVER_READ_RATE

GBL_NFS_SERVER_READ_RATE_CUMGBL_NFS_SERVER_READ_RATE_CUM

GBL_NFS_SERVER_SERVICE_TIMEGBL_NFS_SERVER_SERVICE_TIME

GBL_NFS_SERVER_SERVICE_TIME_CUMGBL_NFS_SERVER_SERVICE_TIME_CUM

GBL_NFS_SERVER_WRITE_BYTE_RATEGBL_NFS_SERVER_WRITE_BYTE_RATE

GBL_NFS_SERVER_WRITE_BYTE_RATE_CUMGBL_NFS_SERVER_WRITE_BYTE_RATE_CUM

GBL_NFS_SERVER_WRITE_RATEGBL_NFS_SERVER_WRITE_RATE

GBL_NFS_SERVER_WRITE_RATE_CUMGBL_NFS_SERVER_WRITE_RATE_CUM

GBL_NFS_WAIT_PCTGBL_NFS_WAIT_PCT

GBL_NFS_WAIT_TIMEGBL_NFS_WAIT_TIME

GBL_NODENAMEGBL_NODENAME

GBL_NUM_ACTIVE_LSGBL_NUM_ACTIVE_LS

GBL_NUM_APPGBL_NUM_APP

GBL_NUM_APP_PRMGBL_NUM_APP_PRM

GBL_NUM_CPUGBL_NUM_CPU

GBL_NUM_CPU_COREGBL_NUM_CPU_CORE

GBL_NUM_DISKGBL_NUM_DISK

GBL_NUM_HBAGBL_NUM_HBA

GBL_NUM_LDOMGBL_NUM_LDOM

GBL_NUM_LSGBL_NUM_LS

GBL_NUM_NETWORKGBL_NUM_NETWORK

GBL_NUM_SOCKETGBL_NUM_SOCKET

GBL_NUM_SWAPGBL_NUM_SWAP

GBL_NUM_TAPEGBL_NUM_TAPE

GBL_NUM_TTGBL_NUM_TT

GBL_NUM_USERGBL_NUM_USER

GBL_NUM_VGGBL_NUM_VG

GBL_NUM_VSWITCHGBL_NUM_VSWITCH

GBL_OSKERNELTYPEGBL_OSKERNELTYPE

GBL_OSKERNELTYPE_INTGBL_OSKERNELTYPE_INT

GBL_OSNAMEGBL_OSNAME

GBL_OSRELEASEGBL_OSRELEASE

GBL_OSVERSIONGBL_OSVERSION

GBL_OTHER_IO_QUEUEGBL_OTHER_IO_QUEUE

GBL_OTHER_IO_WAIT_PCTGBL_OTHER_IO_WAIT_PCT

GBL_OTHER_IO_WAIT_TIMEGBL_OTHER_IO_WAIT_TIME

GBL_OTHER_QUEUEGBL_OTHER_QUEUE

GBL_OTHER_WAIT_PCTGBL_OTHER_WAIT_PCT

GBL_OTHER_WAIT_TIMEGBL_OTHER_WAIT_TIME

GBL_PIPE_QUEUEGBL_PIPE_QUEUE

GBL_PIPE_WAIT_PCTGBL_PIPE_WAIT_PCT

GBL_PIPE_WAIT_TIMEGBL_PIPE_WAIT_TIME

GBL_PRI_QUEUEGBL_PRI_QUEUE

GBL_PRI_WAIT_PCTGBL_PRI_WAIT_PCT

GBL_PRI_WAIT_TIMEGBL_PRI_WAIT_TIME

GBL_PRM_MEM_UTILGBL_PRM_MEM_UTIL

GBL_PROC_RUN_TIMEGBL_PROC_RUN_TIME

GBL_PROC_SAMPLEGBL_PROC_SAMPLE

GBL_RPC_QUEUEGBL_RPC_QUEUE

GBL_RPC_WAIT_PCTGBL_RPC_WAIT_PCT

GBL_RPC_WAIT_TIMEGBL_RPC_WAIT_TIME

GBL_RUN_QUEUEGBL_RUN_QUEUE

GBL_RUN_QUEUE_CUMGBL_RUN_QUEUE_CUM

GBL_RUN_QUEUE_HIGHGBL_RUN_QUEUE_HIGH

GBL_SAMPLEGBL_SAMPLE

GBL_SEM_QUEUEGBL_SEM_QUEUE

GBL_SEM_WAIT_PCTGBL_SEM_WAIT_PCT

GBL_SEM_WAIT_TIMEGBL_SEM_WAIT_TIME

GBL_SERIALNOGBL_SERIALNO

GBL_SLEEP_QUEUEGBL_SLEEP_QUEUE

GBL_SLEEP_WAIT_PCTGBL_SLEEP_WAIT_PCT

GBL_SLEEP_WAIT_TIMEGBL_SLEEP_WAIT_TIME

GBL_SOCKET_QUEUEGBL_SOCKET_QUEUE

GBL_SOCKET_WAIT_PCTGBL_SOCKET_WAIT_PCT

GBL_SOCKET_WAIT_TIMEGBL_SOCKET_WAIT_TIME

GBL_STARTDATEGBL_STARTDATE

GBL_STARTED_PROCGBL_STARTED_PROC

GBL_STARTED_PROC_RATEGBL_STARTED_PROC_RATE

GBL_STARTTIMEGBL_STARTTIME

GBL_STATDATEGBL_STATDATE

GBL_STATTIMEGBL_STATTIME

GBL_STREAM_QUEUEGBL_STREAM_QUEUE

GBL_STREAM_WAIT_PCTGBL_STREAM_WAIT_PCT

GBL_STREAM_WAIT_TIMEGBL_STREAM_WAIT_TIME

GBL_SWAP_RESERVED_ONLY_UTILGBL_SWAP_RESERVED_ONLY_UTIL

GBL_SWAP_SPACE_AVAILGBL_SWAP_SPACE_AVAIL

GBL_SWAP_SPACE_AVAIL_KBGBL_SWAP_SPACE_AVAIL_KB

GBL_SWAP_SPACE_DEVICE_UTILGBL_SWAP_SPACE_DEVICE_UTIL

GBL_SWAP_SPACE_FS_UTILGBL_SWAP_SPACE_FS_UTIL

GBL_SWAP_SPACE_RESERVEDGBL_SWAP_SPACE_RESERVED

GBL_SWAP_SPACE_RESERVED_UTILGBL_SWAP_SPACE_RESERVED_UTIL

GBL_SWAP_SPACE_USEDGBL_SWAP_SPACE_USED

GBL_SWAP_SPACE_USED_UTILGBL_SWAP_SPACE_USED_UTIL

GBL_SWAP_SPACE_UTILGBL_SWAP_SPACE_UTIL

GBL_SWAP_SPACE_UTIL_CUMGBL_SWAP_SPACE_UTIL_CUM

GBL_SWAP_SPACE_UTIL_HIGHGBL_SWAP_SPACE_UTIL_HIGH

GBL_SYSCALLGBL_SYSCALL

GBL_SYSCALL_RATEGBL_SYSCALL_RATE

GBL_SYSCALL_RATE_CUMGBL_SYSCALL_RATE_CUM

GBL_SYSCALL_RATE_HIGHGBL_SYSCALL_RATE_HIGH

GBL_SYSTEM_IDGBL_SYSTEM_ID

GBL_SYSTEM_TYPEGBL_SYSTEM_TYPE

GBL_SYSTEM_UPTIME_HOURSGBL_SYSTEM_UPTIME_HOURS

GBL_SYSTEM_UPTIME_SECONDSGBL_SYSTEM_UPTIME_SECONDS

GBL_SYS_QUEUEGBL_SYS_QUEUE

GBL_SYS_WAIT_PCTGBL_SYS_WAIT_PCT

GBL_SYS_WAIT_TIMEGBL_SYS_WAIT_TIME

GBL_TERM_IO_QUEUEGBL_TERM_IO_QUEUE

GBL_TERM_IO_WAIT_PCTGBL_TERM_IO_WAIT_PCT

GBL_TERM_IO_WAIT_TIMEGBL_TERM_IO_WAIT_TIME

GBL_THRESHOLD_PROCCPUGBL_THRESHOLD_PROCCPU

GBL_THRESHOLD_PROCDISKGBL_THRESHOLD_PROCDISK

GBL_THRESHOLD_PROCIOGBL_THRESHOLD_PROCIO

GBL_THRESHOLD_PROCMEMGBL_THRESHOLD_PROCMEM

GBL_TT_OVERFLOW_COUNTGBL_TT_OVERFLOW_COUNT

Table Metrics

TBL_BUFFER_CACHE_AVAILTBL_BUFFER_CACHE_AVAIL

TBL_BUFFER_CACHE_HIGHTBL_BUFFER_CACHE_HIGH

TBL_BUFFER_CACHE_MAXTBL_BUFFER_CACHE_MAX

TBL_BUFFER_CACHE_MINTBL_BUFFER_CACHE_MIN

TBL_BUFFER_CACHE_USEDTBL_BUFFER_CACHE_USED

TBL_BUFFER_HEADER_AVAILTBL_BUFFER_HEADER_AVAIL

TBL_BUFFER_HEADER_USEDTBL_BUFFER_HEADER_USED

TBL_BUFFER_HEADER_UTILTBL_BUFFER_HEADER_UTIL

TBL_BUFFER_HEADER_UTIL_HIGHTBL_BUFFER_HEADER_UTIL_HIGH

TBL_DNLC_CACHE_AVAILTBL_DNLC_CACHE_AVAIL

TBL_FILE_LOCK_AVAILTBL_FILE_LOCK_AVAIL

TBL_FILE_LOCK_USEDTBL_FILE_LOCK_USED

TBL_FILE_LOCK_UTILTBL_FILE_LOCK_UTIL

TBL_FILE_LOCK_UTIL_HIGHTBL_FILE_LOCK_UTIL_HIGH

TBL_FILE_TABLE_AVAILTBL_FILE_TABLE_AVAIL

TBL_FILE_TABLE_USEDTBL_FILE_TABLE_USED

TBL_FILE_TABLE_UTILTBL_FILE_TABLE_UTIL

TBL_FILE_TABLE_UTIL_HIGHTBL_FILE_TABLE_UTIL_HIGH

TBL_INODE_CACHE_AVAILTBL_INODE_CACHE_AVAIL

TBL_INODE_CACHE_HIGHTBL_INODE_CACHE_HIGH

TBL_INODE_CACHE_USEDTBL_INODE_CACHE_USED

TBL_MSG_BUFFER_AVAILTBL_MSG_BUFFER_AVAIL

TBL_MSG_BUFFER_HIGHTBL_MSG_BUFFER_HIGH

TBL_MSG_BUFFER_USEDTBL_MSG_BUFFER_USED

TBL_MSG_TABLE_AVAILTBL_MSG_TABLE_AVAIL

TBL_MSG_TABLE_USEDTBL_MSG_TABLE_USED

TBL_MSG_TABLE_UTILTBL_MSG_TABLE_UTIL

TBL_MSG_TABLE_UTIL_HIGHTBL_MSG_TABLE_UTIL_HIGH

TBL_PROC_TABLE_AVAILTBL_PROC_TABLE_AVAIL

TBL_PROC_TABLE_USEDTBL_PROC_TABLE_USED

TBL_PROC_TABLE_UTILTBL_PROC_TABLE_UTIL

TBL_PROC_TABLE_UTIL_HIGHTBL_PROC_TABLE_UTIL_HIGH

TBL_PTY_AVAILTBL_PTY_AVAIL

TBL_PTY_USEDTBL_PTY_USED

TBL_PTY_UTILTBL_PTY_UTIL

TBL_PTY_UTIL_HIGHTBL_PTY_UTIL_HIGH

TBL_SEM_TABLE_AVAILTBL_SEM_TABLE_AVAIL

TBL_SEM_TABLE_USEDTBL_SEM_TABLE_USED

TBL_SEM_TABLE_UTILTBL_SEM_TABLE_UTIL

TBL_SEM_TABLE_UTIL_HIGHTBL_SEM_TABLE_UTIL_HIGH

TBL_SHMEM_ACTIVETBL_SHMEM_ACTIVE

TBL_SHMEM_AVAILTBL_SHMEM_AVAIL

TBL_SHMEM_REQUESTEDTBL_SHMEM_REQUESTED

TBL_SHMEM_TABLE_AVAILTBL_SHMEM_TABLE_AVAIL

TBL_SHMEM_TABLE_USEDTBL_SHMEM_TABLE_USED

TBL_SHMEM_TABLE_UTILTBL_SHMEM_TABLE_UTIL

TBL_SHMEM_TABLE_UTIL_HIGHTBL_SHMEM_TABLE_UTIL_HIGH

TBL_SHMEM_USEDTBL_SHMEM_USED

Process Metrics

PROC_APP_IDPROC_APP_ID

PROC_APP_NAMEPROC_APP_NAME

PROC_CACHE_WAIT_PCTPROC_CACHE_WAIT_PCT

PROC_CACHE_WAIT_PCT_CUMPROC_CACHE_WAIT_PCT_CUM

PROC_CACHE_WAIT_TIMEPROC_CACHE_WAIT_TIME

PROC_CACHE_WAIT_TIME_CUMPROC_CACHE_WAIT_TIME_CUM

PROC_CDFS_WAIT_PCTPROC_CDFS_WAIT_PCT

PROC_CDFS_WAIT_PCT_CUMPROC_CDFS_WAIT_PCT_CUM

PROC_CDFS_WAIT_TIMEPROC_CDFS_WAIT_TIME

PROC_CDFS_WAIT_TIME_CUMPROC_CDFS_WAIT_TIME_CUM

PROC_CLOSEPROC_CLOSE

PROC_CLOSE_CUMPROC_CLOSE_CUM

PROC_CPU_ALIVE_SYS_MODE_UTILPROC_CPU_ALIVE_SYS_MODE_UTIL

PROC_CPU_ALIVE_TOTAL_UTILPROC_CPU_ALIVE_TOTAL_UTIL

PROC_CPU_ALIVE_USER_MODE_UTILPROC_CPU_ALIVE_USER_MODE_UTIL

PROC_CPU_CSWITCH_TIMEPROC_CPU_CSWITCH_TIME

PROC_CPU_CSWITCH_TIME_CUMPROC_CPU_CSWITCH_TIME_CUM

PROC_CPU_CSWITCH_UTILPROC_CPU_CSWITCH_UTIL

PROC_CPU_CSWITCH_UTIL_CUMPROC_CPU_CSWITCH_UTIL_CUM

PROC_CPU_INTERRUPT_TIMEPROC_CPU_INTERRUPT_TIME

PROC_CPU_INTERRUPT_TIME_CUMPROC_CPU_INTERRUPT_TIME_CUM

PROC_CPU_INTERRUPT_UTILPROC_CPU_INTERRUPT_UTIL

PROC_CPU_INTERRUPT_UTIL_CUMPROC_CPU_INTERRUPT_UTIL_CUM

PROC_CPU_LAST_USEDPROC_CPU_LAST_USED

PROC_CPU_NICE_TIMEPROC_CPU_NICE_TIME

PROC_CPU_NICE_TIME_CUMPROC_CPU_NICE_TIME_CUM

PROC_CPU_NICE_UTILPROC_CPU_NICE_UTIL

PROC_CPU_NICE_UTIL_CUMPROC_CPU_NICE_UTIL_CUM

PROC_CPU_NNICE_TIMEPROC_CPU_NNICE_TIME

PROC_CPU_NNICE_TIME_CUMPROC_CPU_NNICE_TIME_CUM

PROC_CPU_NNICE_UTILPROC_CPU_NNICE_UTIL

PROC_CPU_NNICE_UTIL_CUMPROC_CPU_NNICE_UTIL_CUM

PROC_CPU_NORMAL_TIMEPROC_CPU_NORMAL_TIME

PROC_CPU_NORMAL_TIME_CUMPROC_CPU_NORMAL_TIME_CUM

PROC_CPU_NORMAL_UTILPROC_CPU_NORMAL_UTIL

PROC_CPU_NORMAL_UTIL_CUMPROC_CPU_NORMAL_UTIL_CUM

PROC_CPU_REALTIME_TIMEPROC_CPU_REALTIME_TIME

PROC_CPU_REALTIME_TIME_CUMPROC_CPU_REALTIME_TIME_CUM

PROC_CPU_REALTIME_UTILPROC_CPU_REALTIME_UTIL

PROC_CPU_REALTIME_UTIL_CUMPROC_CPU_REALTIME_UTIL_CUM

PROC_CPU_SWITCHESPROC_CPU_SWITCHES

PROC_CPU_SWITCHES_CUMPROC_CPU_SWITCHES_CUM

PROC_CPU_SYSCALL_TIMEPROC_CPU_SYSCALL_TIME

PROC_CPU_SYSCALL_TIME_CUMPROC_CPU_SYSCALL_TIME_CUM

PROC_CPU_SYSCALL_UTILPROC_CPU_SYSCALL_UTIL

PROC_CPU_SYSCALL_UTIL_CUMPROC_CPU_SYSCALL_UTIL_CUM

PROC_CPU_SYS_MODE_TIMEPROC_CPU_SYS_MODE_TIME

PROC_CPU_SYS_MODE_TIME_CUMPROC_CPU_SYS_MODE_TIME_CUM

PROC_CPU_SYS_MODE_UTILPROC_CPU_SYS_MODE_UTIL

PROC_CPU_SYS_MODE_UTIL_CUMPROC_CPU_SYS_MODE_UTIL_CUM

PROC_CPU_TOTAL_TIMEPROC_CPU_TOTAL_TIME

PROC_CPU_TOTAL_TIME_CUMPROC_CPU_TOTAL_TIME_CUM

PROC_CPU_TOTAL_UTILPROC_CPU_TOTAL_UTIL

PROC_CPU_TOTAL_UTIL_CUMPROC_CPU_TOTAL_UTIL_CUM

PROC_CPU_TRAP_COUNTPROC_CPU_TRAP_COUNT

PROC_CPU_TRAP_COUNT_CUMPROC_CPU_TRAP_COUNT_CUM

PROC_CPU_USER_MODE_TIMEPROC_CPU_USER_MODE_TIME

PROC_CPU_USER_MODE_TIME_CUMPROC_CPU_USER_MODE_TIME_CUM

PROC_CPU_USER_MODE_UTILPROC_CPU_USER_MODE_UTIL

PROC_CPU_USER_MODE_UTIL_CUMPROC_CPU_USER_MODE_UTIL_CUM

PROC_DISK_FS_READPROC_DISK_FS_READ

PROC_DISK_FS_READ_CUMPROC_DISK_FS_READ_CUM

PROC_DISK_FS_READ_RATEPROC_DISK_FS_READ_RATE

PROC_DISK_FS_WRITEPROC_DISK_FS_WRITE

PROC_DISK_FS_WRITE_CUMPROC_DISK_FS_WRITE_CUM

PROC_DISK_FS_WRITE_RATEPROC_DISK_FS_WRITE_RATE

PROC_DISK_LOGL_IOPROC_DISK_LOGL_IO

PROC_DISK_LOGL_IO_CUMPROC_DISK_LOGL_IO_CUM

PROC_DISK_LOGL_IO_RATEPROC_DISK_LOGL_IO_RATE

PROC_DISK_LOGL_IO_RATE_CUMPROC_DISK_LOGL_IO_RATE_CUM

PROC_DISK_LOGL_READPROC_DISK_LOGL_READ

PROC_DISK_LOGL_READ_CUMPROC_DISK_LOGL_READ_CUM

PROC_DISK_LOGL_READ_RATEPROC_DISK_LOGL_READ_RATE

PROC_DISK_LOGL_WRITEPROC_DISK_LOGL_WRITE

PROC_DISK_LOGL_WRITE_CUMPROC_DISK_LOGL_WRITE_CUM

PROC_DISK_LOGL_WRITE_RATEPROC_DISK_LOGL_WRITE_RATE

PROC_DISK_PHYS_IO_RATEPROC_DISK_PHYS_IO_RATE

PROC_DISK_PHYS_IO_RATE_CUMPROC_DISK_PHYS_IO_RATE_CUM

PROC_DISK_PHYS_READPROC_DISK_PHYS_READ

PROC_DISK_PHYS_READ_CUMPROC_DISK_PHYS_READ_CUM

PROC_DISK_PHYS_READ_RATEPROC_DISK_PHYS_READ_RATE

PROC_DISK_PHYS_WRITEPROC_DISK_PHYS_WRITE

PROC_DISK_PHYS_WRITE_CUMPROC_DISK_PHYS_WRITE_CUM

PROC_DISK_PHYS_WRITE_RATEPROC_DISK_PHYS_WRITE_RATE

PROC_DISK_RAW_READPROC_DISK_RAW_READ

PROC_DISK_RAW_READ_CUMPROC_DISK_RAW_READ_CUM

PROC_DISK_RAW_READ_RATEPROC_DISK_RAW_READ_RATE

PROC_DISK_RAW_WRITEPROC_DISK_RAW_WRITE

PROC_DISK_RAW_WRITE_CUMPROC_DISK_RAW_WRITE_CUM

PROC_DISK_RAW_WRITE_RATEPROC_DISK_RAW_WRITE_RATE

PROC_DISK_REM_LOGL_READPROC_DISK_REM_LOGL_READ

PROC_DISK_REM_LOGL_READ_CUMPROC_DISK_REM_LOGL_READ_CUM

PROC_DISK_REM_LOGL_READ_RATEPROC_DISK_REM_LOGL_READ_RATE

PROC_DISK_REM_LOGL_WRITEPROC_DISK_REM_LOGL_WRITE

PROC_DISK_REM_LOGL_WRITE_CUMPROC_DISK_REM_LOGL_WRITE_CUM

PROC_DISK_REM_LOGL_WRITE_RATEPROC_DISK_REM_LOGL_WRITE_RATE

PROC_DISK_REM_PHYS_READPROC_DISK_REM_PHYS_READ

PROC_DISK_REM_PHYS_READ_CUMPROC_DISK_REM_PHYS_READ_CUM

PROC_DISK_REM_PHYS_READ_RATEPROC_DISK_REM_PHYS_READ_RATE

PROC_DISK_REM_PHYS_WRITEPROC_DISK_REM_PHYS_WRITE

PROC_DISK_REM_PHYS_WRITE_CUMPROC_DISK_REM_PHYS_WRITE_CUM

PROC_DISK_REM_PHYS_WRITE_RATEPROC_DISK_REM_PHYS_WRITE_RATE

PROC_DISK_SUBSYSTEM_WAIT_PCTPROC_DISK_SUBSYSTEM_WAIT_PCT

PROC_DISK_SUBSYSTEM_WAIT_PCT_CUMPROC_DISK_SUBSYSTEM_WAIT_PCT_CUM

PROC_DISK_SUBSYSTEM_WAIT_TIMEPROC_DISK_SUBSYSTEM_WAIT_TIME

PROC_DISK_SUBSYSTEM_WAIT_TIME_CUMPROC_DISK_SUBSYSTEM_WAIT_TIME_
CUM

PROC_DISK_SYSTEM_IOPROC_DISK_SYSTEM_IO

PROC_DISK_SYSTEM_IO_RATEPROC_DISK_SYSTEM_IO_RATE

PROC_DISK_SYSTEM_READPROC_DISK_SYSTEM_READ

PROC_DISK_SYSTEM_READ_CUMPROC_DISK_SYSTEM_READ_CUM

PROC_DISK_SYSTEM_WRITEPROC_DISK_SYSTEM_WRITE

PROC_DISK_SYSTEM_WRITE_CUMPROC_DISK_SYSTEM_WRITE_CUM

PROC_DISK_VM_IOPROC_DISK_VM_IO

PROC_DISK_VM_IO_RATEPROC_DISK_VM_IO_RATE

PROC_DISK_VM_READPROC_DISK_VM_READ

PROC_DISK_VM_READ_CUMPROC_DISK_VM_READ_CUM

PROC_DISK_VM_WRITEPROC_DISK_VM_WRITE

PROC_DISK_VM_WRITE_CUMPROC_DISK_VM_WRITE_CUM

PROC_DISK_WAIT_PCTPROC_DISK_WAIT_PCT

PROC_DISK_WAIT_PCT_CUMPROC_DISK_WAIT_PCT_CUM

PROC_DISK_WAIT_TIMEPROC_DISK_WAIT_TIME

PROC_DISK_WAIT_TIME_CUMPROC_DISK_WAIT_TIME_CUM

PROC_DISPATCHPROC_DISPATCH

PROC_DISPATCH_CUMPROC_DISPATCH_CUM

PROC_EUIDPROC_EUID

PROC_FORCED_CSWITCHPROC_FORCED_CSWITCH

PROC_FORCED_CSWITCH_CUMPROC_FORCED_CSWITCH_CUM

PROC_FORKPROC_FORK

PROC_FORK_CUMPROC_FORK_CUM

PROC_GRAPHICS_WAIT_PCTPROC_GRAPHICS_WAIT_PCT

PROC_GRAPHICS_WAIT_PCT_CUMPROC_GRAPHICS_WAIT_PCT_CUM

PROC_GRAPHICS_WAIT_TIMEPROC_GRAPHICS_WAIT_TIME

PROC_GRAPHICS_WAIT_TIME_CUMPROC_GRAPHICS_WAIT_TIME_CUM

PROC_GROUP_IDPROC_GROUP_ID

PROC_GROUP_NAMEPROC_GROUP_NAME

PROC_INODE_WAIT_PCTPROC_INODE_WAIT_PCT

PROC_INODE_WAIT_PCT_CUMPROC_INODE_WAIT_PCT_CUM

PROC_INODE_WAIT_TIMEPROC_INODE_WAIT_TIME

PROC_INODE_WAIT_TIME_CUMPROC_INODE_WAIT_TIME_CUM

PROC_INTERESTPROC_INTEREST

PROC_INTERRUPTSPROC_INTERRUPTS

PROC_INTERRUPTS_CUMPROC_INTERRUPTS_CUM

PROC_INTERVALPROC_INTERVAL

PROC_INTERVAL_ALIVEPROC_INTERVAL_ALIVE

PROC_INTERVAL_CUMPROC_INTERVAL_CUM

PROC_IOCTLPROC_IOCTL

PROC_IOCTL_CUMPROC_IOCTL_CUM

PROC_IO_BYTEPROC_IO_BYTE

PROC_IO_BYTE_CUMPROC_IO_BYTE_CUM

PROC_IO_BYTE_RATEPROC_IO_BYTE_RATE

PROC_IO_BYTE_RATE_CUMPROC_IO_BYTE_RATE_CUM

PROC_IPC_SUBSYSTEM_WAIT_PCTPROC_IPC_SUBSYSTEM_WAIT_PCT

PROC_IPC_SUBSYSTEM_WAIT_PCT_CUMPROC_IPC_SUBSYSTEM_WAIT_PCT_CUM

PROC_IPC_SUBSYSTEM_WAIT_TIMEPROC_IPC_SUBSYSTEM_WAIT_TIME

PROC_IPC_SUBSYSTEM_WAIT_TIME_CUMPROC_IPC_SUBSYSTEM_WAIT_TIME_CUM

PROC_IPC_WAIT_PCTPROC_IPC_WAIT_PCT

PROC_IPC_WAIT_PCT_CUMPROC_IPC_WAIT_PCT_CUM

PROC_IPC_WAIT_TIMEPROC_IPC_WAIT_TIME

PROC_IPC_WAIT_TIME_CUMPROC_IPC_WAIT_TIME_CUM

PROC_JOBCTL_WAIT_PCTPROC_JOBCTL_WAIT_PCT

PROC_JOBCTL_WAIT_PCT_CUMPROC_JOBCTL_WAIT_PCT_CUM

PROC_JOBCTL_WAIT_TIMEPROC_JOBCTL_WAIT_TIME

PROC_JOBCTL_WAIT_TIME_CUMPROC_JOBCTL_WAIT_TIME_CUM

PROC_LAN_WAIT_PCTPROC_LAN_WAIT_PCT

PROC_LAN_WAIT_PCT_CUMPROC_LAN_WAIT_PCT_CUM

PROC_LAN_WAIT_TIMEPROC_LAN_WAIT_TIME

PROC_LAN_WAIT_TIME_CUMPROC_LAN_WAIT_TIME_CUM

PROC_MAJOR_FAULTPROC_MAJOR_FAULT

PROC_MAJOR_FAULT_CUMPROC_MAJOR_FAULT_CUM

PROC_MEM_PRIVATE_RESPROC_MEM_PRIVATE_RES

PROC_MEM_RESPROC_MEM_RES

PROC_MEM_RES_HIGHPROC_MEM_RES_HIGH

PROC_MEM_SHARED_RESPROC_MEM_SHARED_RES

PROC_MEM_VFAULT_COUNTPROC_MEM_VFAULT_COUNT

PROC_MEM_VFAULT_COUNT_CUMPROC_MEM_VFAULT_COUNT_CUM

PROC_MEM_VIRTPROC_MEM_VIRT

PROC_MEM_WAIT_PCTPROC_MEM_WAIT_PCT

PROC_MEM_WAIT_PCT_CUMPROC_MEM_WAIT_PCT_CUM

PROC_MEM_WAIT_TIMEPROC_MEM_WAIT_TIME

PROC_MEM_WAIT_TIME_CUMPROC_MEM_WAIT_TIME_CUM

PROC_MINOR_FAULTPROC_MINOR_FAULT

PROC_MINOR_FAULT_CUMPROC_MINOR_FAULT_CUM

PROC_MSG_RECEIVEDPROC_MSG_RECEIVED

PROC_MSG_RECEIVED_CUMPROC_MSG_RECEIVED_CUM

PROC_MSG_SENTPROC_MSG_SENT

PROC_MSG_SENT_CUMPROC_MSG_SENT_CUM

PROC_MSG_WAIT_PCTPROC_MSG_WAIT_PCT

PROC_MSG_WAIT_PCT_CUMPROC_MSG_WAIT_PCT_CUM

PROC_MSG_WAIT_TIMEPROC_MSG_WAIT_TIME

PROC_MSG_WAIT_TIME_CUMPROC_MSG_WAIT_TIME_CUM

PROC_NFS_WAIT_PCTPROC_NFS_WAIT_PCT

PROC_NFS_WAIT_PCT_CUMPROC_NFS_WAIT_PCT_CUM

PROC_NFS_WAIT_TIMEPROC_NFS_WAIT_TIME

PROC_NFS_WAIT_TIME_CUMPROC_NFS_WAIT_TIME_CUM

PROC_NICE_PRIPROC_NICE_PRI

PROC_NONDISK_LOGL_READPROC_NONDISK_LOGL_READ

PROC_NONDISK_LOGL_READ_CUMPROC_NONDISK_LOGL_READ_CUM

PROC_NONDISK_LOGL_WRITEPROC_NONDISK_LOGL_WRITE

PROC_NONDISK_LOGL_WRITE_CUMPROC_NONDISK_LOGL_WRITE_CUM

PROC_NONDISK_PHYS_READPROC_NONDISK_PHYS_READ

PROC_NONDISK_PHYS_READ_CUMPROC_NONDISK_PHYS_READ_CUM

PROC_NONDISK_PHYS_WRITEPROC_NONDISK_PHYS_WRITE

PROC_NONDISK_PHYS_WRITE_CUMPROC_NONDISK_PHYS_WRITE_CUM

PROC_OPENPROC_OPEN

PROC_OPEN_CUMPROC_OPEN_CUM

PROC_OTHER_IO_WAIT_PCTPROC_OTHER_IO_WAIT_PCT

PROC_OTHER_IO_WAIT_PCT_CUMPROC_OTHER_IO_WAIT_PCT_CUM

PROC_OTHER_IO_WAIT_TIMEPROC_OTHER_IO_WAIT_TIME

PROC_OTHER_IO_WAIT_TIME_CUMPROC_OTHER_IO_WAIT_TIME_CUM

PROC_OTHER_WAIT_PCTPROC_OTHER_WAIT_PCT

PROC_OTHER_WAIT_PCT_CUMPROC_OTHER_WAIT_PCT_CUM

PROC_OTHER_WAIT_TIMEPROC_OTHER_WAIT_TIME

PROC_OTHER_WAIT_TIME_CUMPROC_OTHER_WAIT_TIME_CUM

PROC_PAGEFAULTPROC_PAGEFAULT

PROC_PAGEFAULT_RATEPROC_PAGEFAULT_RATE

PROC_PAGEFAULT_RATE_CUMPROC_PAGEFAULT_RATE_CUM

PROC_PARENT_PROC_IDPROC_PARENT_PROC_ID

PROC_PIPE_WAIT_PCTPROC_PIPE_WAIT_PCT

PROC_PIPE_WAIT_PCT_CUMPROC_PIPE_WAIT_PCT_CUM

PROC_PIPE_WAIT_TIMEPROC_PIPE_WAIT_TIME

PROC_PIPE_WAIT_TIME_CUMPROC_PIPE_WAIT_TIME_CUM

PROC_PRIPROC_PRI

PROC_PRI_WAIT_PCTPROC_PRI_WAIT_PCT

PROC_PRI_WAIT_PCT_CUMPROC_PRI_WAIT_PCT_CUM

PROC_PRI_WAIT_TIMEPROC_PRI_WAIT_TIME

PROC_PRI_WAIT_TIME_CUMPROC_PRI_WAIT_TIME_CUM

PROC_PRMIDPROC_PRMID

PROC_PROC_ARGV1PROC_PROC_ARGV1

PROC_PROC_CMDPROC_PROC_CMD

PROC_PROC_IDPROC_PROC_ID

PROC_PROC_NAMEPROC_PROC_NAME

PROC_RPC_WAIT_PCTPROC_RPC_WAIT_PCT

PROC_RPC_WAIT_PCT_CUMPROC_RPC_WAIT_PCT_CUM

PROC_RPC_WAIT_TIMEPROC_RPC_WAIT_TIME

PROC_RPC_WAIT_TIME_CUMPROC_RPC_WAIT_TIME_CUM

PROC_RUN_TIMEPROC_RUN_TIME

PROC_SCHEDULERPROC_SCHEDULER

PROC_SEM_WAIT_PCTPROC_SEM_WAIT_PCT

PROC_SEM_WAIT_PCT_CUMPROC_SEM_WAIT_PCT_CUM

PROC_SEM_WAIT_TIMEPROC_SEM_WAIT_TIME

PROC_SEM_WAIT_TIME_CUMPROC_SEM_WAIT_TIME_CUM

PROC_SIGNALPROC_SIGNAL

PROC_SIGNAL_CUMPROC_SIGNAL_CUM

PROC_SLEEP_WAIT_PCTPROC_SLEEP_WAIT_PCT

PROC_SLEEP_WAIT_PCT_CUMPROC_SLEEP_WAIT_PCT_CUM

PROC_SLEEP_WAIT_TIMEPROC_SLEEP_WAIT_TIME

PROC_SLEEP_WAIT_TIME_CUMPROC_SLEEP_WAIT_TIME_CUM

PROC_SOCKET_WAIT_PCTPROC_SOCKET_WAIT_PCT

PROC_SOCKET_WAIT_PCT_CUMPROC_SOCKET_WAIT_PCT_CUM

PROC_SOCKET_WAIT_TIMEPROC_SOCKET_WAIT_TIME

PROC_SOCKET_WAIT_TIME_CUMPROC_SOCKET_WAIT_TIME_CUM

PROC_STARTTIMEPROC_STARTTIME

PROC_STATEPROC_STATE

PROC_STOP_REASONPROC_STOP_REASON

PROC_STOP_REASON_FLAGPROC_STOP_REASON_FLAG

PROC_STREAM_WAIT_PCTPROC_STREAM_WAIT_PCT

PROC_STREAM_WAIT_PCT_CUMPROC_STREAM_WAIT_PCT_CUM

PROC_STREAM_WAIT_TIMEPROC_STREAM_WAIT_TIME

PROC_STREAM_WAIT_TIME_CUMPROC_STREAM_WAIT_TIME_CUM

PROC_SWAPPROC_SWAP

PROC_SWAP_CUMPROC_SWAP_CUM

PROC_SYS_WAIT_PCTPROC_SYS_WAIT_PCT

PROC_SYS_WAIT_PCT_CUMPROC_SYS_WAIT_PCT_CUM

PROC_SYS_WAIT_TIMEPROC_SYS_WAIT_TIME

PROC_SYS_WAIT_TIME_CUMPROC_SYS_WAIT_TIME_CUM

PROC_TERM_IO_WAIT_PCTPROC_TERM_IO_WAIT_PCT

PROC_TERM_IO_WAIT_PCT_CUMPROC_TERM_IO_WAIT_PCT_CUM

PROC_TERM_IO_WAIT_TIMEPROC_TERM_IO_WAIT_TIME

PROC_TERM_IO_WAIT_TIME_CUMPROC_TERM_IO_WAIT_TIME_CUM

PROC_THREAD_COUNTPROC_THREAD_COUNT

PROC_THREAD_IDPROC_THREAD_ID

PROC_TIMEPROC_TIME

PROC_TOP_CPU_INDEXPROC_TOP_CPU_INDEX

PROC_TOP_DISK_INDEXPROC_TOP_DISK_INDEX

PROC_TOTAL_WAIT_TIMEPROC_TOTAL_WAIT_TIME

PROC_TOTAL_WAIT_TIME_CUMPROC_TOTAL_WAIT_TIME_CUM

PROC_TTYPROC_TTY

PROC_TTY_DEVPROC_TTY_DEV

PROC_UIDPROC_UID

PROC_USER_NAMEPROC_USER_NAME

PROC_USER_THREAD_IDPROC_USER_THREAD_ID

PROC_USRPRIPROC_USRPRI

PROC_VOLUNTARY_CSWITCHPROC_VOLUNTARY_CSWITCH

PROC_VOLUNTARY_CSWITCH_CUMPROC_VOLUNTARY_CSWITCH_CUM

Application Metrics

APP_ACTIVE_APPAPP_ACTIVE_APP

APP_ACTIVE_APP_PRMAPP_ACTIVE_APP_PRM

APP_ACTIVE_PROCAPP_ACTIVE_PROC

APP_ALIVE_PROCAPP_ALIVE_PROC

APP_COMPLETED_PROCAPP_COMPLETED_PROC

APP_CPU_NICE_TIMEAPP_CPU_NICE_TIME

APP_CPU_NICE_UTILAPP_CPU_NICE_UTIL

APP_CPU_NNICE_TIMEAPP_CPU_NNICE_TIME

APP_CPU_NNICE_UTILAPP_CPU_NNICE_UTIL

APP_CPU_NORMAL_TIMEAPP_CPU_NORMAL_TIME

APP_CPU_NORMAL_UTILAPP_CPU_NORMAL_UTIL

APP_CPU_REALTIME_TIMEAPP_CPU_REALTIME_TIME

APP_CPU_REALTIME_UTILAPP_CPU_REALTIME_UTIL

APP_CPU_SYS_MODE_TIMEAPP_CPU_SYS_MODE_TIME

APP_CPU_SYS_MODE_UTILAPP_CPU_SYS_MODE_UTIL

APP_CPU_TOTAL_TIMEAPP_CPU_TOTAL_TIME

APP_CPU_TOTAL_UTILAPP_CPU_TOTAL_UTIL

APP_CPU_TOTAL_UTIL_CUMAPP_CPU_TOTAL_UTIL_CUM

APP_CPU_USER_MODE_TIMEAPP_CPU_USER_MODE_TIME

APP_CPU_USER_MODE_UTILAPP_CPU_USER_MODE_UTIL

APP_DISK_FS_IO_RATEAPP_DISK_FS_IO_RATE

APP_DISK_LOGL_IO_RATEAPP_DISK_LOGL_IO_RATE

APP_DISK_LOGL_READAPP_DISK_LOGL_READ

APP_DISK_LOGL_READ_RATEAPP_DISK_LOGL_READ_RATE

APP_DISK_LOGL_WRITEAPP_DISK_LOGL_WRITE

APP_DISK_LOGL_WRITE_RATEAPP_DISK_LOGL_WRITE_RATE

APP_DISK_PHYS_IO_RATEAPP_DISK_PHYS_IO_RATE

APP_DISK_PHYS_READAPP_DISK_PHYS_READ

APP_DISK_PHYS_READ_RATEAPP_DISK_PHYS_READ_RATE

APP_DISK_PHYS_WRITEAPP_DISK_PHYS_WRITE

APP_DISK_PHYS_WRITE_RATEAPP_DISK_PHYS_WRITE_RATE

APP_DISK_RAW_IO_RATEAPP_DISK_RAW_IO_RATE

APP_DISK_SUBSYSTEM_QUEUEAPP_DISK_SUBSYSTEM_QUEUE

APP_DISK_SUBSYSTEM_WAIT_PCTAPP_DISK_SUBSYSTEM_WAIT_PCT

APP_DISK_SYSTEM_IO_RATEAPP_DISK_SYSTEM_IO_RATE

APP_DISK_VM_IO_RATEAPP_DISK_VM_IO_RATE

APP_INTERVALAPP_INTERVAL

APP_INTERVAL_CUMAPP_INTERVAL_CUM

APP_IO_BYTEAPP_IO_BYTE

APP_IO_BYTE_RATEAPP_IO_BYTE_RATE

APP_IPC_SUBSYSTEM_QUEUEAPP_IPC_SUBSYSTEM_QUEUE

APP_IPC_SUBSYSTEM_WAIT_PCTAPP_IPC_SUBSYSTEM_WAIT_PCT

APP_MAJOR_FAULTAPP_MAJOR_FAULT

APP_MAJOR_FAULT_RATEAPP_MAJOR_FAULT_RATE

APP_MEM_QUEUEAPP_MEM_QUEUE

APP_MEM_RESAPP_MEM_RES

APP_MEM_UTILAPP_MEM_UTIL

APP_MEM_VIRTAPP_MEM_VIRT

APP_MEM_WAIT_PCTAPP_MEM_WAIT_PCT

APP_MINOR_FAULTAPP_MINOR_FAULT

APP_MINOR_FAULT_RATEAPP_MINOR_FAULT_RATE

APP_NAMEAPP_NAME

APP_NAME_PRM_GROUPNAMEAPP_NAME_PRM_GROUPNAME

APP_NETWORK_SUBSYSTEM_QUEUEAPP_NETWORK_SUBSYSTEM_QUEUE

APP_NETWORK_SUBSYSTEM_WAIT_PCTAPP_NETWORK_SUBSYSTEM_WAIT_PCT

APP_NUMAPP_NUM

APP_OTHER_IO_QUEUEAPP_OTHER_IO_QUEUE

APP_OTHER_IO_WAIT_PCTAPP_OTHER_IO_WAIT_PCT

APP_PRIAPP_PRI

APP_PRI_QUEUEAPP_PRI_QUEUE

APP_PRI_STD_DEVAPP_PRI_STD_DEV

APP_PRI_WAIT_PCTAPP_PRI_WAIT_PCT

APP_PRM_CPUCAP_MODEAPP_PRM_CPUCAP_MODE

APP_PRM_CPU_ENTITLEMENTAPP_PRM_CPU_ENTITLEMENT

APP_PRM_CPU_TOTAL_UTIL_CUMAPP_PRM_CPU_TOTAL_UTIL_CUM

APP_PRM_DISK_STATEAPP_PRM_DISK_STATE

APP_PRM_GROUPIDAPP_PRM_GROUPID

APP_PRM_INTERVAL_CUMAPP_PRM_INTERVAL_CUM

APP_PRM_MEM_AVAILAPP_PRM_MEM_AVAIL

APP_PRM_MEM_ENTITLEMENTAPP_PRM_MEM_ENTITLEMENT

APP_PRM_MEM_STATEAPP_PRM_MEM_STATE

APP_PRM_MEM_UPPERBOUNDAPP_PRM_MEM_UPPERBOUND

APP_PRM_MEM_UTILAPP_PRM_MEM_UTIL

APP_PRM_STATEAPP_PRM_STATE

APP_PRM_SUSPENDED_PROCAPP_PRM_SUSPENDED_PROC

APP_PROC_RUN_TIMEAPP_PROC_RUN_TIME

APP_SAMPLEAPP_SAMPLE

APP_SEM_QUEUEAPP_SEM_QUEUE

APP_SEM_WAIT_PCTAPP_SEM_WAIT_PCT

APP_SLEEP_QUEUEAPP_SLEEP_QUEUE

APP_SLEEP_WAIT_PCTAPP_SLEEP_WAIT_PCT

APP_TERM_IO_QUEUEAPP_TERM_IO_QUEUE

APP_TERM_IO_WAIT_PCTAPP_TERM_IO_WAIT_PCT

APP_TIMEAPP_TIME

Process By File Metrics

PROC_FILE_COUNTPROC_FILE_COUNT

PROC_FILE_MODEPROC_FILE_MODE

PROC_FILE_NAMEPROC_FILE_NAME

PROC_FILE_NUMBERPROC_FILE_NUMBER

PROC_FILE_OFFSETPROC_FILE_OFFSET

PROC_FILE_OPENPROC_FILE_OPEN

PROC_FILE_TYPEPROC_FILE_TYPE

By Disk Metrics

BYDSK_AVG_QUEUE_TIMEBYDSK_AVG_QUEUE_TIME

BYDSK_AVG_READ_QUEUE_TIMEBYDSK_AVG_READ_QUEUE_TIME

BYDSK_AVG_READ_SERVICE_TIMEBYDSK_AVG_READ_SERVICE_TIME

BYDSK_AVG_SERVICE_TIMEBYDSK_AVG_SERVICE_TIME

BYDSK_AVG_WRITE_QUEUE_TIMEBYDSK_AVG_WRITE_QUEUE_TIME

BYDSK_AVG_WRITE_SERVICE_TIMEBYDSK_AVG_WRITE_SERVICE_TIME

BYDSK_BUSBYDSK_BUS

BYDSK_BUSY_TIMEBYDSK_BUSY_TIME

BYDSK_CONTROLLERBYDSK_CONTROLLER

BYDSK_DEVNAMEBYDSK_DEVNAME

BYDSK_DEVNOBYDSK_DEVNO

BYDSK_DIRNAMEBYDSK_DIRNAME

BYDSK_DISKNAMEBYDSK_DISKNAME

BYDSK_FS_IO_RATEBYDSK_FS_IO_RATE

BYDSK_FS_READBYDSK_FS_READ

BYDSK_FS_READ_RATEBYDSK_FS_READ_RATE

BYDSK_FS_WRITEBYDSK_FS_WRITE

BYDSK_FS_WRITE_RATEBYDSK_FS_WRITE_RATE

BYDSK_IDBYDSK_ID

BYDSK_INTERVALBYDSK_INTERVAL

BYDSK_INTERVAL_CUMBYDSK_INTERVAL_CUM

BYDSK_LOGL_BYTE_RATEBYDSK_LOGL_BYTE_RATE

BYDSK_LOGL_BYTE_RATE_CUMBYDSK_LOGL_BYTE_RATE_CUM

BYDSK_LOGL_IO_RATEBYDSK_LOGL_IO_RATE

BYDSK_LOGL_IO_RATE_CUMBYDSK_LOGL_IO_RATE_CUM

BYDSK_LOGL_READBYDSK_LOGL_READ

BYDSK_LOGL_READ_BYTE_RATEBYDSK_LOGL_READ_BYTE_RATE

BYDSK_LOGL_READ_BYTE_RATE_CUMBYDSK_LOGL_READ_BYTE_RATE_CUM

BYDSK_LOGL_READ_RATEBYDSK_LOGL_READ_RATE

BYDSK_LOGL_READ_RATE_CUMBYDSK_LOGL_READ_RATE_CUM

BYDSK_LOGL_WRITEBYDSK_LOGL_WRITE

BYDSK_LOGL_WRITE_BYTE_RATEBYDSK_LOGL_WRITE_BYTE_RATE

BYDSK_LOGL_WRITE_BYTE_RATE_CUMBYDSK_LOGL_WRITE_BYTE_RATE_CUM

BYDSK_LOGL_WRITE_RATEBYDSK_LOGL_WRITE_RATE

BYDSK_LOGL_WRITE_RATE_CUMBYDSK_LOGL_WRITE_RATE_CUM

BYDSK_PHYS_BYTEBYDSK_PHYS_BYTE

BYDSK_PHYS_BYTE_RATEBYDSK_PHYS_BYTE_RATE

BYDSK_PHYS_BYTE_RATE_CUMBYDSK_PHYS_BYTE_RATE_CUM

BYDSK_PHYS_IOBYDSK_PHYS_IO

BYDSK_PHYS_IO_RATEBYDSK_PHYS_IO_RATE

BYDSK_PHYS_IO_RATE_CUMBYDSK_PHYS_IO_RATE_CUM

BYDSK_PHYS_READBYDSK_PHYS_READ

BYDSK_PHYS_READ_BYTEBYDSK_PHYS_READ_BYTE

BYDSK_PHYS_READ_BYTE_RATEBYDSK_PHYS_READ_BYTE_RATE

BYDSK_PHYS_READ_BYTE_RATE_CUMBYDSK_PHYS_READ_BYTE_RATE_CUM

BYDSK_PHYS_READ_RATEBYDSK_PHYS_READ_RATE

BYDSK_PHYS_READ_RATE_CUMBYDSK_PHYS_READ_RATE_CUM

BYDSK_PHYS_WRITEBYDSK_PHYS_WRITE

BYDSK_PHYS_WRITE_BYTEBYDSK_PHYS_WRITE_BYTE

BYDSK_PHYS_WRITE_BYTE_RATEBYDSK_PHYS_WRITE_BYTE_RATE

BYDSK_PHYS_WRITE_BYTE_RATE_CUMBYDSK_PHYS_WRITE_BYTE_RATE_CUM

BYDSK_PHYS_WRITE_RATEBYDSK_PHYS_WRITE_RATE

BYDSK_PHYS_WRITE_RATE_CUMBYDSK_PHYS_WRITE_RATE_CUM

BYDSK_PRODUCT_IDBYDSK_PRODUCT_ID

BYDSK_QUEUE_0_UTILBYDSK_QUEUE_0_UTIL

BYDSK_QUEUE_2_UTILBYDSK_QUEUE_2_UTIL

BYDSK_QUEUE_4_UTILBYDSK_QUEUE_4_UTIL

BYDSK_QUEUE_8_UTILBYDSK_QUEUE_8_UTIL

BYDSK_QUEUE_X_UTILBYDSK_QUEUE_X_UTIL

BYDSK_RAW_IO_RATEBYDSK_RAW_IO_RATE

BYDSK_RAW_READBYDSK_RAW_READ

BYDSK_RAW_READ_RATEBYDSK_RAW_READ_RATE

BYDSK_RAW_WRITEBYDSK_RAW_WRITE

BYDSK_RAW_WRITE_RATEBYDSK_RAW_WRITE_RATE

BYDSK_REQUEST_QUEUEBYDSK_REQUEST_QUEUE

BYDSK_SYSTEM_IOBYDSK_SYSTEM_IO

BYDSK_SYSTEM_IO_RATEBYDSK_SYSTEM_IO_RATE

BYDSK_SYSTEM_READ_RATEBYDSK_SYSTEM_READ_RATE

BYDSK_SYSTEM_WRITE_RATEBYDSK_SYSTEM_WRITE_RATE

BYDSK_TIMEBYDSK_TIME

BYDSK_UTILBYDSK_UTIL

BYDSK_UTIL_CUMBYDSK_UTIL_CUM

BYDSK_VENDOR_IDBYDSK_VENDOR_ID

BYDSK_VM_IOBYDSK_VM_IO

BYDSK_VM_IO_RATEBYDSK_VM_IO_RATE

BYDSK_VM_READ_RATEBYDSK_VM_READ_RATE

BYDSK_VM_WRITE_RATEBYDSK_VM_WRITE_RATE

File System Metrics

FS_BLOCK_SIZEFS_BLOCK_SIZE

FS_DEVNAMEFS_DEVNAME

FS_DEVNOFS_DEVNO

FS_DIRNAMEFS_DIRNAME

FS_FILE_IO_RATEFS_FILE_IO_RATE

FS_FILE_IO_RATE_CUMFS_FILE_IO_RATE_CUM

FS_FRAG_SIZEFS_FRAG_SIZE

FS_INODE_UTILFS_INODE_UTIL

FS_INTERVALFS_INTERVAL

FS_INTERVAL_CUMFS_INTERVAL_CUM

FS_IS_LVMFS_IS_LVM

FS_LOGL_IO_RATEFS_LOGL_IO_RATE

FS_LOGL_IO_RATE_CUMFS_LOGL_IO_RATE_CUM

FS_LOGL_READ_BYTE_RATEFS_LOGL_READ_BYTE_RATE

FS_LOGL_READ_BYTE_RATE_CUMFS_LOGL_READ_BYTE_RATE_CUM

FS_LOGL_READ_RATEFS_LOGL_READ_RATE

FS_LOGL_READ_RATE_CUMFS_LOGL_READ_RATE_CUM

FS_LOGL_WRITE_BYTE_RATEFS_LOGL_WRITE_BYTE_RATE

FS_LOGL_WRITE_BYTE_RATE_CUMFS_LOGL_WRITE_BYTE_RATE_CUM

FS_LOGL_WRITE_RATEFS_LOGL_WRITE_RATE

FS_LOGL_WRITE_RATE_CUMFS_LOGL_WRITE_RATE_CUM

FS_MAX_INODESFS_MAX_INODES

FS_MAX_SIZEFS_MAX_SIZE

FS_PHYS_IO_RATEFS_PHYS_IO_RATE

FS_PHYS_IO_RATE_CUMFS_PHYS_IO_RATE_CUM

FS_PHYS_READ_BYTE_RATEFS_PHYS_READ_BYTE_RATE

FS_PHYS_READ_BYTE_RATE_CUMFS_PHYS_READ_BYTE_RATE_CUM

FS_PHYS_READ_RATEFS_PHYS_READ_RATE

FS_PHYS_READ_RATE_CUMFS_PHYS_READ_RATE_CUM

FS_PHYS_WRITE_BYTE_RATEFS_PHYS_WRITE_BYTE_RATE

FS_PHYS_WRITE_BYTE_RATE_CUMFS_PHYS_WRITE_BYTE_RATE_CUM

FS_PHYS_WRITE_RATEFS_PHYS_WRITE_RATE

FS_PHYS_WRITE_RATE_CUMFS_PHYS_WRITE_RATE_CUM

FS_SPACE_RESERVEDFS_SPACE_RESERVED

FS_SPACE_USEDFS_SPACE_USED

FS_SPACE_UTILFS_SPACE_UTIL

FS_TYPEFS_TYPE

FS_VM_IO_RATEFS_VM_IO_RATE

FS_VM_IO_RATE_CUMFS_VM_IO_RATE_CUM

Logical Volume Metrics

LV_AVG_READ_SERVICE_TIMELV_AVG_READ_SERVICE_TIME

LV_AVG_WRITE_SERVICE_TIMELV_AVG_WRITE_SERVICE_TIME

LV_CACHE_HITLV_CACHE_HIT

LV_CACHE_MISSLV_CACHE_MISS

LV_CACHE_QUEUELV_CACHE_QUEUE

LV_CACHE_SIZELV_CACHE_SIZE

LV_DEVNOLV_DEVNO

LV_DIRNAMELV_DIRNAME

LV_GROUP_NAMELV_GROUP_NAME

LV_INTERVALLV_INTERVAL

LV_INTERVAL_CUMLV_INTERVAL_CUM

LV_OPEN_LVLV_OPEN_LV

LV_READ_BYTE_RATELV_READ_BYTE_RATE

LV_READ_BYTE_RATE_CUMLV_READ_BYTE_RATE_CUM

LV_READ_RATELV_READ_RATE

LV_READ_RATE_CUMLV_READ_RATE_CUM

LV_TYPELV_TYPE

LV_WRITE_BYTE_RATELV_WRITE_BYTE_RATE

LV_WRITE_BYTE_RATE_CUMLV_WRITE_BYTE_RATE_CUM

LV_WRITE_RATELV_WRITE_RATE

LV_WRITE_RATE_CUMLV_WRITE_RATE_CUM

By Network Interface Metrics

BYNETIF_COLLISIONBYNETIF_COLLISION

BYNETIF_COLLISION_1_MIN_RATEBYNETIF_COLLISION_1_MIN_RATE

BYNETIF_COLLISION_RATEBYNETIF_COLLISION_RATE

BYNETIF_COLLISION_RATE_CUMBYNETIF_COLLISION_RATE_CUM

BYNETIF_ERRORBYNETIF_ERROR

BYNETIF_ERROR_1_MIN_RATEBYNETIF_ERROR_1_MIN_RATE

BYNETIF_ERROR_RATEBYNETIF_ERROR_RATE

BYNETIF_ERROR_RATE_CUMBYNETIF_ERROR_RATE_CUM

BYNETIF_IDBYNETIF_ID

BYNETIF_INTERVALBYNETIF_INTERVAL

BYNETIF_INTERVAL_CUMBYNETIF_INTERVAL_CUM

BYNETIF_IN_BYTEBYNETIF_IN_BYTE

BYNETIF_IN_BYTE_RATEBYNETIF_IN_BYTE_RATE

BYNETIF_IN_BYTE_RATE_CUMBYNETIF_IN_BYTE_RATE_CUM

BYNETIF_IN_PACKETBYNETIF_IN_PACKET

BYNETIF_IN_PACKET_RATEBYNETIF_IN_PACKET_RATE

BYNETIF_IN_PACKET_RATE_CUMBYNETIF_IN_PACKET_RATE_CUM

BYNETIF_NAMEBYNETIF_NAME

BYNETIF_NET_MTUBYNETIF_NET_MTU

BYNETIF_NET_SPEEDBYNETIF_NET_SPEED

BYNETIF_NET_TYPEBYNETIF_NET_TYPE

BYNETIF_OUT_BYTEBYNETIF_OUT_BYTE

BYNETIF_OUT_BYTE_RATEBYNETIF_OUT_BYTE_RATE

BYNETIF_OUT_BYTE_RATE_CUMBYNETIF_OUT_BYTE_RATE_CUM

BYNETIF_OUT_PACKETBYNETIF_OUT_PACKET

BYNETIF_OUT_PACKET_RATEBYNETIF_OUT_PACKET_RATE

BYNETIF_OUT_PACKET_RATE_CUMBYNETIF_OUT_PACKET_RATE_CUM

BYNETIF_PACKET_RATEBYNETIF_PACKET_RATE

BYNETIF_QUEUEBYNETIF_QUEUE

BYNETIF_UTILBYNETIF_UTIL

By Swap Metrics

BYSWP_SWAP_PRIBYSWP_SWAP_PRI

BYSWP_SWAP_SPACE_AVAILBYSWP_SWAP_SPACE_AVAIL

BYSWP_SWAP_SPACE_NAMEBYSWP_SWAP_SPACE_NAME

BYSWP_SWAP_SPACE_USEDBYSWP_SWAP_SPACE_USED

BYSWP_SWAP_TYPEBYSWP_SWAP_TYPE

By CPU Metrics

BYCPU_ACTIVEBYCPU_ACTIVE

BYCPU_CPU_CLOCKBYCPU_CPU_CLOCK

BYCPU_CPU_CSWITCH_TIMEBYCPU_CPU_CSWITCH_TIME

BYCPU_CPU_CSWITCH_TIME_CUMBYCPU_CPU_CSWITCH_TIME_CUM

BYCPU_CPU_CSWITCH_UTILBYCPU_CPU_CSWITCH_UTIL

BYCPU_CPU_CSWITCH_UTIL_CUMBYCPU_CPU_CSWITCH_UTIL_CUM

BYCPU_CPU_INTERRUPT_TIMEBYCPU_CPU_INTERRUPT_TIME

BYCPU_CPU_INTERRUPT_TIME_CUMBYCPU_CPU_INTERRUPT_TIME_CUM

BYCPU_CPU_INTERRUPT_UTILBYCPU_CPU_INTERRUPT_UTIL

BYCPU_CPU_INTERRUPT_UTIL_CUMBYCPU_CPU_INTERRUPT_UTIL_CUM

BYCPU_CPU_NICE_TIMEBYCPU_CPU_NICE_TIME

BYCPU_CPU_NICE_TIME_CUMBYCPU_CPU_NICE_TIME_CUM

BYCPU_CPU_NICE_UTILBYCPU_CPU_NICE_UTIL

BYCPU_CPU_NICE_UTIL_CUMBYCPU_CPU_NICE_UTIL_CUM

BYCPU_CPU_NNICE_TIMEBYCPU_CPU_NNICE_TIME

BYCPU_CPU_NNICE_TIME_CUMBYCPU_CPU_NNICE_TIME_CUM

BYCPU_CPU_NNICE_UTILBYCPU_CPU_NNICE_UTIL

BYCPU_CPU_NNICE_UTIL_CUMBYCPU_CPU_NNICE_UTIL_CUM

BYCPU_CPU_NORMAL_TIMEBYCPU_CPU_NORMAL_TIME

BYCPU_CPU_NORMAL_TIME_CUMBYCPU_CPU_NORMAL_TIME_CUM

BYCPU_CPU_NORMAL_UTILBYCPU_CPU_NORMAL_UTIL

BYCPU_CPU_NORMAL_UTIL_CUMBYCPU_CPU_NORMAL_UTIL_CUM

BYCPU_CPU_REALTIME_TIMEBYCPU_CPU_REALTIME_TIME

BYCPU_CPU_REALTIME_TIME_CUMBYCPU_CPU_REALTIME_TIME_CUM

BYCPU_CPU_REALTIME_UTILBYCPU_CPU_REALTIME_UTIL

BYCPU_CPU_REALTIME_UTIL_CUMBYCPU_CPU_REALTIME_UTIL_CUM

BYCPU_CPU_SYSCALL_TIMEBYCPU_CPU_SYSCALL_TIME

BYCPU_CPU_SYSCALL_TIME_CUMBYCPU_CPU_SYSCALL_TIME_CUM

BYCPU_CPU_SYSCALL_UTILBYCPU_CPU_SYSCALL_UTIL

BYCPU_CPU_SYSCALL_UTIL_CUMBYCPU_CPU_SYSCALL_UTIL_CUM

BYCPU_CPU_SYS_MODE_TIMEBYCPU_CPU_SYS_MODE_TIME

BYCPU_CPU_SYS_MODE_TIME_CUMBYCPU_CPU_SYS_MODE_TIME_CUM

BYCPU_CPU_SYS_MODE_UTILBYCPU_CPU_SYS_MODE_UTIL

BYCPU_CPU_SYS_MODE_UTIL_CUMBYCPU_CPU_SYS_MODE_UTIL_CUM

BYCPU_CPU_TOTAL_TIMEBYCPU_CPU_TOTAL_TIME

BYCPU_CPU_TOTAL_TIME_CUMBYCPU_CPU_TOTAL_TIME_CUM

BYCPU_CPU_TOTAL_UTILBYCPU_CPU_TOTAL_UTIL

BYCPU_CPU_TOTAL_UTIL_CUMBYCPU_CPU_TOTAL_UTIL_CUM

BYCPU_CPU_TRAP_TIMEBYCPU_CPU_TRAP_TIME

BYCPU_CPU_TRAP_TIME_CUMBYCPU_CPU_TRAP_TIME_CUM

BYCPU_CPU_TRAP_UTILBYCPU_CPU_TRAP_UTIL

BYCPU_CPU_TRAP_UTIL_CUMBYCPU_CPU_TRAP_UTIL_CUM

BYCPU_CPU_USER_MODE_TIMEBYCPU_CPU_USER_MODE_TIME

BYCPU_CPU_USER_MODE_TIME_CUMBYCPU_CPU_USER_MODE_TIME_CUM

BYCPU_CPU_USER_MODE_UTILBYCPU_CPU_USER_MODE_UTIL

BYCPU_CPU_USER_MODE_UTIL_CUMBYCPU_CPU_USER_MODE_UTIL_CUM

BYCPU_CPU_VFAULT_TIMEBYCPU_CPU_VFAULT_TIME

BYCPU_CPU_VFAULT_TIME_CUMBYCPU_CPU_VFAULT_TIME_CUM

BYCPU_CPU_VFAULT_UTILBYCPU_CPU_VFAULT_UTIL

BYCPU_CPU_VFAULT_UTIL_CUMBYCPU_CPU_VFAULT_UTIL_CUM

BYCPU_CSWITCHBYCPU_CSWITCH

BYCPU_CSWITCH_CUMBYCPU_CSWITCH_CUM

BYCPU_CSWITCH_RATEBYCPU_CSWITCH_RATE

BYCPU_CSWITCH_RATE_CUMBYCPU_CSWITCH_RATE_CUM

BYCPU_IDBYCPU_ID

BYCPU_INTERRUPTBYCPU_INTERRUPT

BYCPU_INTERRUPT_RATEBYCPU_INTERRUPT_RATE

BYCPU_INTERRUPT_STATEBYCPU_INTERRUPT_STATE

BYCPU_LAST_PROC_IDBYCPU_LAST_PROC_ID

BYCPU_LAST_THREAD_IDBYCPU_LAST_THREAD_ID

BYCPU_LAST_USER_THREAD_IDBYCPU_LAST_USER_THREAD_ID

BYCPU_RUN_QUEUE_15_MINBYCPU_RUN_QUEUE_15_MIN

BYCPU_RUN_QUEUE_1_MINBYCPU_RUN_QUEUE_1_MIN

BYCPU_RUN_QUEUE_5_MINBYCPU_RUN_QUEUE_5_MIN

BYCPU_STATEBYCPU_STATE

Process By Memory Region Metrics

PROC_REGION_FILENAMEPROC_REGION_FILENAME

PROC_REGION_LOCKEDPROC_REGION_LOCKED

PROC_REGION_PAGE_COUNT_1_4KBPROC_REGION_PAGE_COUNT_1_4KB

PROC_REGION_PAGE_COUNT_2_16KBPROC_REGION_PAGE_COUNT_2_16KB

PROC_REGION_PAGE_COUNT_3_64KBPROC_REGION_PAGE_COUNT_3_64KB

PROC_REGION_PAGE_COUNT_4_256KBPROC_REGION_PAGE_COUNT_4_256KB

PROC_REGION_PAGE_COUNT_5_1MBPROC_REGION_PAGE_COUNT_5_1MB

PROC_REGION_PAGE_COUNT_6_4MBPROC_REGION_PAGE_COUNT_6_4MB

PROC_REGION_PAGE_COUNT_7_16MBPROC_REGION_PAGE_COUNT_7_16MB

PROC_REGION_PAGE_COUNT_8_64MBPROC_REGION_PAGE_COUNT_8_64MB

PROC_REGION_PAGE_COUNT_9_256MBPROC_REGION_PAGE_COUNT_9_256MB

PROC_REGION_PAGE_COUNT_B_1GBPROC_REGION_PAGE_COUNT_B_1GB

PROC_REGION_PAGE_COUNT_B_4GBPROC_REGION_PAGE_COUNT_B_4GB

PROC_REGION_PAGE_SIZE_HINTPROC_REGION_PAGE_SIZE_HINT

PROC_REGION_PRIVATE_SHARED_FLAGPROC_REGION_PRIVATE_SHARED_FLAG

PROC_REGION_REF_COUNTPROC_REGION_REF_COUNT

PROC_REGION_RESPROC_REGION_RES

PROC_REGION_RES_DATAPROC_REGION_RES_DATA

PROC_REGION_RES_OTHERPROC_REGION_RES_OTHER

PROC_REGION_RES_SHMEMPROC_REGION_RES_SHMEM

PROC_REGION_RES_STACKPROC_REGION_RES_STACK

PROC_REGION_RES_TEXTPROC_REGION_RES_TEXT

PROC_REGION_TYPEPROC_REGION_TYPE

PROC_REGION_VIRTPROC_REGION_VIRT

PROC_REGION_VIRT_ADDRSPROC_REGION_VIRT_ADDRS

PROC_REGION_VIRT_DATAPROC_REGION_VIRT_DATA

PROC_REGION_VIRT_OTHERPROC_REGION_VIRT_OTHER

PROC_REGION_VIRT_SHMEMPROC_REGION_VIRT_SHMEM

PROC_REGION_VIRT_STACKPROC_REGION_VIRT_STACK

PROC_REGION_VIRT_TEXTPROC_REGION_VIRT_TEXT

By NFS Metrics

BYNFS_CLIENT_PHYS_TIMEBYNFS_CLIENT_PHYS_TIME

BYNFS_CLIENT_PHYS_TIME_CUMBYNFS_CLIENT_PHYS_TIME_CUM

BYNFS_CLIENT_READ_BYTE_RATEBYNFS_CLIENT_READ_BYTE_RATE

BYNFS_CLIENT_READ_BYTE_RATE_CUMBYNFS_CLIENT_READ_BYTE_RATE_CUM

BYNFS_CLIENT_READ_RATEBYNFS_CLIENT_READ_RATE

BYNFS_CLIENT_READ_RATE_CUMBYNFS_CLIENT_READ_RATE_CUM

BYNFS_CLIENT_SERVICEBYNFS_CLIENT_SERVICE

BYNFS_CLIENT_SERVICE_CUMBYNFS_CLIENT_SERVICE_CUM

BYNFS_CLIENT_SERVICE_QUEUEBYNFS_CLIENT_SERVICE_QUEUE

BYNFS_CLIENT_SERVICE_QUEUE_CUMBYNFS_CLIENT_SERVICE_QUEUE_CUM

BYNFS_CLIENT_SERVICE_TIMEBYNFS_CLIENT_SERVICE_TIME

BYNFS_CLIENT_SERVICE_TIME_CUMBYNFS_CLIENT_SERVICE_TIME_CUM

BYNFS_CLIENT_WRITE_BYTE_RATEBYNFS_CLIENT_WRITE_BYTE_RATE

BYNFS_CLIENT_WRITE_BYTE_RATE_CUMBYNFS_CLIENT_WRITE_BYTE_RATE_CUM

BYNFS_CLIENT_WRITE_RATEBYNFS_CLIENT_WRITE_RATE

BYNFS_CLIENT_WRITE_RATE_CUMBYNFS_CLIENT_WRITE_RATE_CUM

BYNFS_HOSTNAMEBYNFS_HOSTNAME

BYNFS_HOST_IP_ADDRESSBYNFS_HOST_IP_ADDRESS

BYNFS_INTERVALBYNFS_INTERVAL

BYNFS_INTERVAL_CUMBYNFS_INTERVAL_CUM

BYNFS_LAST_PROC_IDBYNFS_LAST_PROC_ID

BYNFS_SERVER_READ_BYTE_RATEBYNFS_SERVER_READ_BYTE_RATE

BYNFS_SERVER_READ_BYTE_RATE_CUMBYNFS_SERVER_READ_BYTE_RATE_CUM

BYNFS_SERVER_READ_RATEBYNFS_SERVER_READ_RATE

BYNFS_SERVER_READ_RATE_CUMBYNFS_SERVER_READ_RATE_CUM

BYNFS_SERVER_SERVICEBYNFS_SERVER_SERVICE

BYNFS_SERVER_SERVICE_CUMBYNFS_SERVER_SERVICE_CUM

BYNFS_SERVER_SERVICE_TIMEBYNFS_SERVER_SERVICE_TIME

BYNFS_SERVER_SERVICE_TIME_CUMBYNFS_SERVER_SERVICE_TIME_CUM

BYNFS_SERVER_WRITE_BYTE_RATEBYNFS_SERVER_WRITE_BYTE_RATE

BYNFS_SERVER_WRITE_BYTE_RATE_CUMBYNFS_SERVER_WRITE_BYTE_RATE_CUM

BYNFS_SERVER_WRITE_RATEBYNFS_SERVER_WRITE_RATE

BYNFS_SERVER_WRITE_RATE_CUMBYNFS_SERVER_WRITE_RATE_CUM

By NFS Operation Metrics

BYNFSOP_CLIENT_COUNTBYNFSOP_CLIENT_COUNT

BYNFSOP_CLIENT_COUNT_CUMBYNFSOP_CLIENT_COUNT_CUM

BYNFSOP_CLIENT_TIMEBYNFSOP_CLIENT_TIME

BYNFSOP_CLIENT_TIME_CUMBYNFSOP_CLIENT_TIME_CUM

BYNFSOP_INTERVALBYNFSOP_INTERVAL

BYNFSOP_INTERVAL_CUMBYNFSOP_INTERVAL_CUM

BYNFSOP_NAMEBYNFSOP_NAME

BYNFSOP_SERVER_COUNTBYNFSOP_SERVER_COUNT

BYNFSOP_SERVER_COUNT_CUMBYNFSOP_SERVER_COUNT_CUM

BYNFSOP_SERVER_TIMEBYNFSOP_SERVER_TIME

BYNFSOP_SERVER_TIME_CUMBYNFSOP_SERVER_TIME_CUM

By Operation Metrics

BYOP_CLIENT_COUNTBYOP_CLIENT_COUNT

BYOP_CLIENT_COUNT_CUMBYOP_CLIENT_COUNT_CUM

BYOP_INTERVALBYOP_INTERVAL

BYOP_INTERVAL_CUMBYOP_INTERVAL_CUM

BYOP_NAMEBYOP_NAME

BYOP_SERVER_COUNTBYOP_SERVER_COUNT

BYOP_SERVER_COUNT_CUMBYOP_SERVER_COUNT_CUM

System Call Metrics

SYSCALL_ACTIVE_CUMSYSCALL_ACTIVE_CUM

SYSCALL_CALL_COUNTSYSCALL_CALL_COUNT

SYSCALL_CALL_COUNT_CUMSYSCALL_CALL_COUNT_CUM

SYSCALL_CALL_IDSYSCALL_CALL_ID

SYSCALL_CALL_NAMESYSCALL_CALL_NAME

SYSCALL_CALL_RATESYSCALL_CALL_RATE

SYSCALL_CALL_RATE_CUMSYSCALL_CALL_RATE_CUM

SYSCALL_CPU_TOTAL_TIMESYSCALL_CPU_TOTAL_TIME

SYSCALL_CPU_TOTAL_TIME_CUMSYSCALL_CPU_TOTAL_TIME_CUM

SYSCALL_INTERVALSYSCALL_INTERVAL

SYSCALL_INTERVAL_CUMSYSCALL_INTERVAL_CUM

By Disk Detail Metrics

BYDSKDETAIL_LABELBYDSKDETAIL_LABEL

BYDSKDETAIL_NAMEBYDSKDETAIL_NAME

File System Detail Metrics

FSDETAIL_LABELFSDETAIL_LABEL

FSDETAIL_NAMEFSDETAIL_NAME

Logical Volume Detail Metrics

LVDETAIL_LABELLVDETAIL_LABEL

LVDETAIL_NAMELVDETAIL_NAME

Transaction Metrics

TT_ABORTTT_ABORT

TT_ABORT_CUMTT_ABORT_CUM

TT_ABORT_WALL_TIMETT_ABORT_WALL_TIME

TT_ABORT_WALL_TIME_CUMTT_ABORT_WALL_TIME_CUM

TT_APPNOTT_APPNO

TT_APP_NAMETT_APP_NAME

TT_CACHE_WAIT_TIME_PER_TRANTT_CACHE_WAIT_TIME_PER_TRAN

TT_CACHE_WAIT_TIME_PER_TRAN_CUMTT_CACHE_WAIT_TIME_PER_TRAN_CUM

TT_CDFS_WAIT_TIME_PER_TRANTT_CDFS_WAIT_TIME_PER_TRAN

TT_CDFS_WAIT_TIME_PER_TRAN_CUMTT_CDFS_WAIT_TIME_PER_TRAN_CUM

TT_CLIENT_CORRELATOR_COUNTTT_CLIENT_CORRELATOR_COUNT

TT_COUNTTT_COUNT

TT_COUNT_CUMTT_COUNT_CUM

TT_CPU_CSWITCH_TIME_PER_TRANTT_CPU_CSWITCH_TIME_PER_TRAN

TT_CPU_CSWITCH_TIME_PER_TRAN_CUMTT_CPU_CSWITCH_TIME_PER_TRAN_CUM

TT_CPU_INTERRUPT_TIME_PER_TRANTT_CPU_INTERRUPT_TIME_PER_TRAN

TT_CPU_INTERRUPT_TIME_PER_TRAN_CUMTT_CPU_INTERRUPT_TIME_PER_TRAN_
CUM

TT_CPU_NICE_TIME_PER_TRANTT_CPU_NICE_TIME_PER_TRAN

TT_CPU_NICE_TIME_PER_TRAN_CUMTT_CPU_NICE_TIME_PER_TRAN_CUM

TT_CPU_NNICE_TIME_PER_TRANTT_CPU_NNICE_TIME_PER_TRAN

TT_CPU_NNICE_TIME_PER_TRAN_CUMTT_CPU_NNICE_TIME_PER_TRAN_CUM

TT_CPU_NORMAL_TIME_PER_TRANTT_CPU_NORMAL_TIME_PER_TRAN

TT_CPU_NORMAL_TIME_PER_TRAN_CUMTT_CPU_NORMAL_TIME_PER_TRAN_CUM

TT_CPU_REALTIME_TIME_PER_TRANTT_CPU_REALTIME_TIME_PER_TRAN

TT_CPU_REALTIME_TIME_PER_TRAN_CUMTT_CPU_REALTIME_TIME_PER_TRAN_CUM

TT_CPU_SYSCALL_TIME_PER_TRANTT_CPU_SYSCALL_TIME_PER_TRAN

TT_CPU_SYSCALL_TIME_PER_TRAN_CUMTT_CPU_SYSCALL_TIME_PER_TRAN_CUM

TT_CPU_SYS_MODE_TIME_PER_TRANTT_CPU_SYS_MODE_TIME_PER_TRAN

TT_CPU_SYS_MODE_TIME_PER_TRAN_CUMTT_CPU_SYS_MODE_TIME_PER_TRAN_
CUM

TT_CPU_TOTAL_TIME_PER_TRANTT_CPU_TOTAL_TIME_PER_TRAN

TT_CPU_TOTAL_TIME_PER_TRAN_CUMTT_CPU_TOTAL_TIME_PER_TRAN_CUM

TT_CPU_USER_MODE_TIME_PER_TRANTT_CPU_USER_MODE_TIME_PER_TRAN

TT_CPU_USER_MODE_TIME_PER_TRAN_CUMTT_CPU_USER_MODE_TIME_PER_TRAN_CUM

TT_DISK_FS_READ_PER_TRANTT_DISK_FS_READ_PER_TRAN

TT_DISK_FS_READ_PER_TRAN_CUMTT_DISK_FS_READ_PER_TRAN_CUM

TT_DISK_FS_WRITE_PER_TRANTT_DISK_FS_WRITE_PER_TRAN

TT_DISK_FS_WRITE_PER_TRAN_CUMTT_DISK_FS_WRITE_PER_TRAN_CUM

TT_DISK_LOGL_IO_PER_TRANTT_DISK_LOGL_IO_PER_TRAN

TT_DISK_LOGL_IO_PER_TRAN_CUMTT_DISK_LOGL_IO_PER_TRAN_CUM

TT_DISK_LOGL_READ_PER_TRANTT_DISK_LOGL_READ_PER_TRAN

TT_DISK_LOGL_READ_PER_TRAN_CUMTT_DISK_LOGL_READ_PER_TRAN_CUM

TT_DISK_LOGL_WRITE_PER_TRANTT_DISK_LOGL_WRITE_PER_TRAN

TT_DISK_LOGL_WRITE_PER_TRAN_CUMTT_DISK_LOGL_WRITE_PER_TRAN_CUM

TT_DISK_PHYS_IO_PER_TRANTT_DISK_PHYS_IO_PER_TRAN

TT_DISK_PHYS_IO_PER_TRAN_CUMTT_DISK_PHYS_IO_PER_TRAN_CUM

TT_DISK_PHYS_READ_PER_TRANTT_DISK_PHYS_READ_PER_TRAN

TT_DISK_PHYS_READ_PER_TRAN_CUMTT_DISK_PHYS_READ_PER_TRAN_CUM

TT_DISK_PHYS_WRITE_PER_TRANTT_DISK_PHYS_WRITE_PER_TRAN

TT_DISK_PHYS_WRITE_PER_TRAN_CUMTT_DISK_PHYS_WRITE_PER_TRAN_CUM

TT_DISK_RAW_READ_PER_TRANTT_DISK_RAW_READ_PER_TRAN

TT_DISK_RAW_READ_PER_TRAN_CUMTT_DISK_RAW_READ_PER_TRAN_CUM

TT_DISK_RAW_WRITE_PER_TRANTT_DISK_RAW_WRITE_PER_TRAN

TT_DISK_RAW_WRITE_PER_TRAN_CUMTT_DISK_RAW_WRITE_PER_TRAN_CUM

TT_DISK_SYSTEM_READ_PER_TRANTT_DISK_SYSTEM_READ_PER_TRAN

TT_DISK_SYSTEM_READ_PER_TRAN_CUMTT_DISK_SYSTEM_READ_PER_TRAN_CUM

TT_DISK_SYSTEM_WRITE_PER_TRANTT_DISK_SYSTEM_WRITE_PER_TRAN

TT_DISK_SYSTEM_WRITE_PER_TRAN_CUMTT_DISK_SYSTEM_WRITE_PER_TRAN_CUM

TT_DISK_VM_READ_PER_TRANTT_DISK_VM_READ_PER_TRAN

TT_DISK_VM_READ_PER_TRAN_CUMTT_DISK_VM_READ_PER_TRAN_CUM

TT_DISK_VM_WRITE_PER_TRANTT_DISK_VM_WRITE_PER_TRAN

TT_DISK_VM_WRITE_PER_TRAN_CUMTT_DISK_VM_WRITE_PER_TRAN_CUM

TT_DISK_WAIT_TIME_PER_TRANTT_DISK_WAIT_TIME_PER_TRAN

TT_DISK_WAIT_TIME_PER_TRAN_CUMTT_DISK_WAIT_TIME_PER_TRAN_CUM

TT_FAILEDTT_FAILED

TT_FAILED_CUMTT_FAILED_CUM

TT_FAILED_WALL_TIMETT_FAILED_WALL_TIME

TT_FAILED_WALL_TIME_CUMTT_FAILED_WALL_TIME_CUM

TT_GOLDENRESOURCE_INTERVALTT_GOLDENRESOURCE_INTERVAL

TT_GOLDENRESOURCE_INTERVAL_CUMTT_GOLDENRESOURCE_INTERVAL_CUM

TT_GRAPHICS_WAIT_TIME_PER_TRANTT_GRAPHICS_WAIT_TIME_PER_TRAN

TT_GRAPHICS_WAIT_TIME_PER_TRAN_CUMTT_GRAPHICS_WAIT_TIME_PER_TRAN_
CUM

TT_INFOTT_INFO

TT_INODE_WAIT_TIME_PER_TRANTT_INODE_WAIT_TIME_PER_TRAN

TT_INODE_WAIT_TIME_PER_TRAN_CUMTT_INODE_WAIT_TIME_PER_TRAN_CUM

TT_INPROGRESS_COUNTTT_INPROGRESS_COUNT

TT_INTERVALTT_INTERVAL

TT_INTERVAL_CUMTT_INTERVAL_CUM

TT_IPC_WAIT_TIME_PER_TRANTT_IPC_WAIT_TIME_PER_TRAN

TT_IPC_WAIT_TIME_PER_TRAN_CUMTT_IPC_WAIT_TIME_PER_TRAN_CUM

TT_JOBCTL_WAIT_TIME_PER_TRANTT_JOBCTL_WAIT_TIME_PER_TRAN

TT_JOBCTL_WAIT_TIME_PER_TRAN_CUMTT_JOBCTL_WAIT_TIME_PER_TRAN_CUM

TT_LAN_WAIT_TIME_PER_TRANTT_LAN_WAIT_TIME_PER_TRAN

TT_LAN_WAIT_TIME_PER_TRAN_CUMTT_LAN_WAIT_TIME_PER_TRAN_CUM

TT_MEASUREMENT_COUNTTT_MEASUREMENT_COUNT

TT_MEM_WAIT_TIME_PER_TRANTT_MEM_WAIT_TIME_PER_TRAN

TT_MEM_WAIT_TIME_PER_TRAN_CUMTT_MEM_WAIT_TIME_PER_TRAN_CUM

TT_MSG_WAIT_TIME_PER_TRANTT_MSG_WAIT_TIME_PER_TRAN

TT_MSG_WAIT_TIME_PER_TRAN_CUMTT_MSG_WAIT_TIME_PER_TRAN_CUM

TT_NAMETT_NAME

TT_NFS_WAIT_TIME_PER_TRANTT_NFS_WAIT_TIME_PER_TRAN

TT_NFS_WAIT_TIME_PER_TRAN_CUMTT_NFS_WAIT_TIME_PER_TRAN_CUM

TT_OTHER_IO_WAIT_TIME_PER_TRANTT_OTHER_IO_WAIT_TIME_PER_TRAN

TT_OTHER_IO_WAIT_TIME_PER_TRAN_CUMTT_OTHER_IO_WAIT_TIME_PER_TRAN_CUM

TT_OTHER_WAIT_TIME_PER_TRANTT_OTHER_WAIT_TIME_PER_TRAN

TT_OTHER_WAIT_TIME_PER_TRAN_CUMTT_OTHER_WAIT_TIME_PER_TRAN_CUM

TT_PIPE_WAIT_TIME_PER_TRANTT_PIPE_WAIT_TIME_PER_TRAN

TT_PIPE_WAIT_TIME_PER_TRAN_CUMTT_PIPE_WAIT_TIME_PER_TRAN_CUM

TT_PRI_WAIT_TIME_PER_TRANTT_PRI_WAIT_TIME_PER_TRAN

TT_PRI_WAIT_TIME_PER_TRAN_CUMTT_PRI_WAIT_TIME_PER_TRAN_CUM

TT_RESOURCE_INTERVALTT_RESOURCE_INTERVAL

TT_RESOURCE_INTERVAL_CUMTT_RESOURCE_INTERVAL_CUM

TT_RPC_WAIT_TIME_PER_TRANTT_RPC_WAIT_TIME_PER_TRAN

TT_RPC_WAIT_TIME_PER_TRAN_CUMTT_RPC_WAIT_TIME_PER_TRAN_CUM

TT_SEM_WAIT_TIME_PER_TRANTT_SEM_WAIT_TIME_PER_TRAN

TT_SEM_WAIT_TIME_PER_TRAN_CUMTT_SEM_WAIT_TIME_PER_TRAN_CUM

TT_SLEEP_WAIT_TIME_PER_TRANTT_SLEEP_WAIT_TIME_PER_TRAN

TT_SLEEP_WAIT_TIME_PER_TRAN_CUMTT_SLEEP_WAIT_TIME_PER_TRAN_CUM

TT_SLO_COUNTTT_SLO_COUNT

TT_SLO_COUNT_CUMTT_SLO_COUNT_CUM

TT_SLO_PERCENTTT_SLO_PERCENT

TT_SLO_THRESHOLDTT_SLO_THRESHOLD

TT_SOCKET_WAIT_TIME_PER_TRANTT_SOCKET_WAIT_TIME_PER_TRAN

TT_SOCKET_WAIT_TIME_PER_TRAN_CUMTT_SOCKET_WAIT_TIME_PER_TRAN_CUM

TT_STREAM_WAIT_TIME_PER_TRANTT_STREAM_WAIT_TIME_PER_TRAN

TT_STREAM_WAIT_TIME_PER_TRAN_CUMTT_STREAM_WAIT_TIME_PER_TRAN_CUM

TT_SYS_WAIT_TIME_PER_TRANTT_SYS_WAIT_TIME_PER_TRAN

TT_SYS_WAIT_TIME_PER_TRAN_CUMTT_SYS_WAIT_TIME_PER_TRAN_CUM

TT_TERM_IO_WAIT_TIME_PER_TRANTT_TERM_IO_WAIT_TIME_PER_TRAN

TT_TERM_IO_WAIT_TIME_PER_TRAN_CUMTT_TERM_IO_WAIT_TIME_PER_TRAN_CUM

TT_TOTAL_WAIT_TIME_PER_TRANTT_TOTAL_WAIT_TIME_PER_TRAN

TT_TOTAL_WAIT_TIME_PER_TRAN_CUMTT_TOTAL_WAIT_TIME_PER_TRAN_CUM

TT_TRAN_1_MIN_RATETT_TRAN_1_MIN_RATE

TT_TRAN_IDTT_TRAN_ID

TT_UNAMETT_UNAME

TT_UPDATETT_UPDATE

TT_UPDATE_CUMTT_UPDATE_CUM

TT_WALL_TIMETT_WALL_TIME

TT_WALL_TIME_CUMTT_WALL_TIME_CUM

TT_WALL_TIME_PER_TRANTT_WALL_TIME_PER_TRAN

TT_WALL_TIME_PER_TRAN_CUMTT_WALL_TIME_PER_TRAN_CUM

Transaction Measurement Section Metrics

TTBIN_TRANS_COUNTTTBIN_TRANS_COUNT

TTBIN_TRANS_COUNT_CUMTTBIN_TRANS_COUNT_CUM

TTBIN_UPPER_RANGETTBIN_UPPER_RANGE

By Process System Call Metrics

PROCSYSCALL_ACTIVE_CUMPROCSYSCALL_ACTIVE_CUM

PROCSYSCALL_CALL_COUNTPROCSYSCALL_CALL_COUNT

PROCSYSCALL_CALL_COUNT_CUMPROCSYSCALL_CALL_COUNT_CUM

PROCSYSCALL_CALL_IDPROCSYSCALL_CALL_ID

PROCSYSCALL_CALL_NAMEPROCSYSCALL_CALL_NAME

PROCSYSCALL_CALL_RATEPROCSYSCALL_CALL_RATE

PROCSYSCALL_CALL_RATE_CUMPROCSYSCALL_CALL_RATE_CUM

PROCSYSCALL_INTERVALPROCSYSCALL_INTERVAL

PROCSYSCALL_INTERVAL_CUMPROCSYSCALL_INTERVAL_CUM

PROCSYSCALL_TOTAL_TIMEPROCSYSCALL_TOTAL_TIME

PROCSYSCALL_TOTAL_TIME_CUMPROCSYSCALL_TOTAL_TIME_CUM

Thread Metrics

THREAD_APP_IDPROC_APP_ID

THREAD_APP_NAMEPROC_APP_NAME

THREAD_CACHE_WAIT_PCTPROC_CACHE_WAIT_PCT

THREAD_CACHE_WAIT_PCT_CUMPROC_CACHE_WAIT_PCT_CUM

THREAD_CACHE_WAIT_TIMEPROC_CACHE_WAIT_TIME

THREAD_CACHE_WAIT_TIME_CUMPROC_CACHE_WAIT_TIME_CUM

THREAD_CDFS_WAIT_PCTPROC_CDFS_WAIT_PCT

THREAD_CDFS_WAIT_PCT_CUMPROC_CDFS_WAIT_PCT_CUM

THREAD_CDFS_WAIT_TIMEPROC_CDFS_WAIT_TIME

THREAD_CDFS_WAIT_TIME_CUMPROC_CDFS_WAIT_TIME_CUM

THREAD_CLOSEPROC_CLOSE

THREAD_CLOSE_CUMPROC_CLOSE_CUM

THREAD_CPU_ALIVE_SYS_MODE_UTILPROC_CPU_ALIVE_SYS_MODE_UTIL

THREAD_CPU_ALIVE_TOTAL_UTILPROC_CPU_ALIVE_TOTAL_UTIL

THREAD_CPU_ALIVE_USER_MODE_UTILPROC_CPU_ALIVE_USER_MODE_UTIL

THREAD_CPU_CSWITCH_TIMEPROC_CPU_CSWITCH_TIME

THREAD_CPU_CSWITCH_TIME_CUMPROC_CPU_CSWITCH_TIME_CUM

THREAD_CPU_CSWITCH_UTILPROC_CPU_CSWITCH_UTIL

THREAD_CPU_CSWITCH_UTIL_CUMPROC_CPU_CSWITCH_UTIL_CUM

THREAD_CPU_INTERRUPT_TIMEPROC_CPU_INTERRUPT_TIME

THREAD_CPU_INTERRUPT_TIME_CUMPROC_CPU_INTERRUPT_TIME_CUM

THREAD_CPU_INTERRUPT_UTILPROC_CPU_INTERRUPT_UTIL

THREAD_CPU_INTERRUPT_UTIL_CUMPROC_CPU_INTERRUPT_UTIL_CUM

THREAD_CPU_LAST_USEDPROC_CPU_LAST_USED

THREAD_CPU_NICE_TIMEPROC_CPU_NICE_TIME

THREAD_CPU_NICE_TIME_CUMPROC_CPU_NICE_TIME_CUM

THREAD_CPU_NICE_UTILPROC_CPU_NICE_UTIL

THREAD_CPU_NICE_UTIL_CUMPROC_CPU_NICE_UTIL_CUM

THREAD_CPU_NNICE_TIMEPROC_CPU_NNICE_TIME

THREAD_CPU_NNICE_TIME_CUMPROC_CPU_NNICE_TIME_CUM

THREAD_CPU_NNICE_UTILPROC_CPU_NNICE_UTIL

THREAD_CPU_NNICE_UTIL_CUMPROC_CPU_NNICE_UTIL_CUM

THREAD_CPU_NORMAL_TIMEPROC_CPU_NORMAL_TIME

THREAD_CPU_NORMAL_TIME_CUMPROC_CPU_NORMAL_TIME_CUM

THREAD_CPU_NORMAL_UTILPROC_CPU_NORMAL_UTIL

THREAD_CPU_NORMAL_UTIL_CUMPROC_CPU_NORMAL_UTIL_CUM

THREAD_CPU_REALTIME_TIMEPROC_CPU_REALTIME_TIME

THREAD_CPU_REALTIME_TIME_CUMPROC_CPU_REALTIME_TIME_CUM

THREAD_CPU_REALTIME_UTILPROC_CPU_REALTIME_UTIL

THREAD_CPU_REALTIME_UTIL_CUMPROC_CPU_REALTIME_UTIL_CUM

THREAD_CPU_SWITCHESPROC_CPU_SWITCHES

THREAD_CPU_SWITCHES_CUMPROC_CPU_SWITCHES_CUM

THREAD_CPU_SYSCALL_TIMEPROC_CPU_SYSCALL_TIME

THREAD_CPU_SYSCALL_TIME_CUMPROC_CPU_SYSCALL_TIME_CUM

THREAD_CPU_SYSCALL_UTILPROC_CPU_SYSCALL_UTIL

THREAD_CPU_SYSCALL_UTIL_CUMPROC_CPU_SYSCALL_UTIL_CUM

THREAD_CPU_SYS_MODE_TIMEPROC_CPU_SYS_MODE_TIME

THREAD_CPU_SYS_MODE_TIME_CUMPROC_CPU_SYS_MODE_TIME_CUM

THREAD_CPU_SYS_MODE_UTILPROC_CPU_SYS_MODE_UTIL

THREAD_CPU_SYS_MODE_UTIL_CUMPROC_CPU_SYS_MODE_UTIL_CUM

THREAD_CPU_TOTAL_TIMEPROC_CPU_TOTAL_TIME

THREAD_CPU_TOTAL_TIME_CUMPROC_CPU_TOTAL_TIME_CUM

THREAD_CPU_TOTAL_UTILPROC_CPU_TOTAL_UTIL

THREAD_CPU_TOTAL_UTIL_CUMPROC_CPU_TOTAL_UTIL_CUM

THREAD_CPU_TRAP_COUNTPROC_CPU_TRAP_COUNT

THREAD_CPU_TRAP_COUNT_CUMPROC_CPU_TRAP_COUNT_CUM

THREAD_CPU_USER_MODE_TIMEPROC_CPU_USER_MODE_TIME

THREAD_CPU_USER_MODE_TIME_CUMPROC_CPU_USER_MODE_TIME_CUM

THREAD_CPU_USER_MODE_UTILPROC_CPU_USER_MODE_UTIL

THREAD_CPU_USER_MODE_UTIL_CUMPROC_CPU_USER_MODE_UTIL_CUM

THREAD_DISK_FS_READPROC_DISK_FS_READ

THREAD_DISK_FS_READ_CUMPROC_DISK_FS_READ_CUM

THREAD_DISK_FS_READ_RATEPROC_DISK_FS_READ_RATE

THREAD_DISK_FS_WRITEPROC_DISK_FS_WRITE

THREAD_DISK_FS_WRITE_CUMPROC_DISK_FS_WRITE_CUM

THREAD_DISK_FS_WRITE_RATEPROC_DISK_FS_WRITE_RATE

THREAD_DISK_LOGL_IOPROC_DISK_LOGL_IO

THREAD_DISK_LOGL_IO_CUMPROC_DISK_LOGL_IO_CUM

THREAD_DISK_LOGL_IO_RATEPROC_DISK_LOGL_IO_RATE

THREAD_DISK_LOGL_IO_RATE_CUMPROC_DISK_LOGL_IO_RATE_CUM

THREAD_DISK_LOGL_READPROC_DISK_LOGL_READ

THREAD_DISK_LOGL_READ_CUMPROC_DISK_LOGL_READ_CUM

THREAD_DISK_LOGL_READ_RATEPROC_DISK_LOGL_READ_RATE

THREAD_DISK_LOGL_WRITEPROC_DISK_LOGL_WRITE

THREAD_DISK_LOGL_WRITE_CUMPROC_DISK_LOGL_WRITE_CUM

THREAD_DISK_LOGL_WRITE_RATEPROC_DISK_LOGL_WRITE_RATE

THREAD_DISK_PHYS_IO_RATEPROC_DISK_PHYS_IO_RATE

THREAD_DISK_PHYS_IO_RATE_CUMPROC_DISK_PHYS_IO_RATE_CUM

THREAD_DISK_PHYS_READPROC_DISK_PHYS_READ

THREAD_DISK_PHYS_READ_CUMPROC_DISK_PHYS_READ_CUM

THREAD_DISK_PHYS_READ_RATEPROC_DISK_PHYS_READ_RATE

THREAD_DISK_PHYS_WRITEPROC_DISK_PHYS_WRITE

THREAD_DISK_PHYS_WRITE_CUMPROC_DISK_PHYS_WRITE_CUM

THREAD_DISK_PHYS_WRITE_RATEPROC_DISK_PHYS_WRITE_RATE

THREAD_DISK_RAW_READPROC_DISK_RAW_READ

THREAD_DISK_RAW_READ_CUMPROC_DISK_RAW_READ_CUM

THREAD_DISK_RAW_READ_RATEPROC_DISK_RAW_READ_RATE

THREAD_DISK_RAW_WRITEPROC_DISK_RAW_WRITE

THREAD_DISK_RAW_WRITE_CUMPROC_DISK_RAW_WRITE_CUM

THREAD_DISK_RAW_WRITE_RATEPROC_DISK_RAW_WRITE_RATE

THREAD_DISK_REM_LOGL_READPROC_DISK_REM_LOGL_READ

THREAD_DISK_REM_LOGL_READ_CUMPROC_DISK_REM_LOGL_READ_CUM

THREAD_DISK_REM_LOGL_READ_RATEPROC_DISK_REM_LOGL_READ_RATE

THREAD_DISK_REM_LOGL_WRITEPROC_DISK_REM_LOGL_WRITE

THREAD_DISK_REM_LOGL_WRITE_CUMPROC_DISK_REM_LOGL_WRITE_CUM

THREAD_DISK_REM_LOGL_WRITE_RATEPROC_DISK_REM_LOGL_WRITE_RATE

THREAD_DISK_REM_PHYS_READPROC_DISK_REM_PHYS_READ

THREAD_DISK_REM_PHYS_READ_CUMPROC_DISK_REM_PHYS_READ_CUM

THREAD_DISK_REM_PHYS_READ_RATEPROC_DISK_REM_PHYS_READ_RATE

THREAD_DISK_REM_PHYS_WRITEPROC_DISK_REM_PHYS_WRITE

THREAD_DISK_REM_PHYS_WRITE_CUMPROC_DISK_REM_PHYS_WRITE_CUM

THREAD_DISK_REM_PHYS_WRITE_RATEPROC_DISK_REM_PHYS_WRITE_RATE

THREAD_DISK_SUBSYSTEM_WAIT_PCTPROC_DISK_SUBSYSTEM_WAIT_PCT

THREAD_DISK_SUBSYSTEM_WAIT_PCT_CUMPROC_DISK_SUBSYSTEM_WAIT_PCT_
CUM

THREAD_DISK_SUBSYSTEM_WAIT_TIMEPROC_DISK_SUBSYSTEM_WAIT_TIME

THREAD_DISK_SUBSYSTEM_WAIT_TIME_CUMPROC_DISK_SUBSYSTEM_WAIT_TIME_
CUM

THREAD_DISK_SYSTEM_IOPROC_DISK_SYSTEM_IO

THREAD_DISK_SYSTEM_IO_RATEPROC_DISK_SYSTEM_IO_RATE

THREAD_DISK_SYSTEM_READPROC_DISK_SYSTEM_READ

THREAD_DISK_SYSTEM_READ_CUMPROC_DISK_SYSTEM_READ_CUM

THREAD_DISK_SYSTEM_WRITEPROC_DISK_SYSTEM_WRITE

THREAD_DISK_SYSTEM_WRITE_CUMPROC_DISK_SYSTEM_WRITE_CUM

THREAD_DISK_VM_IOPROC_DISK_VM_IO

THREAD_DISK_VM_IO_RATEPROC_DISK_VM_IO_RATE

THREAD_DISK_VM_READPROC_DISK_VM_READ

THREAD_DISK_VM_READ_CUMPROC_DISK_VM_READ_CUM

THREAD_DISK_VM_WRITEPROC_DISK_VM_WRITE

THREAD_DISK_VM_WRITE_CUMPROC_DISK_VM_WRITE_CUM

THREAD_DISK_WAIT_PCTPROC_DISK_WAIT_PCT

THREAD_DISK_WAIT_PCT_CUMPROC_DISK_WAIT_PCT_CUM

THREAD_DISK_WAIT_TIMEPROC_DISK_WAIT_TIME

THREAD_DISK_WAIT_TIME_CUMPROC_DISK_WAIT_TIME_CUM

THREAD_DISPATCHPROC_DISPATCH

THREAD_DISPATCH_CUMPROC_DISPATCH_CUM

THREAD_EUIDPROC_EUID

THREAD_FORCED_CSWITCHPROC_FORCED_CSWITCH

THREAD_FORCED_CSWITCH_CUMPROC_FORCED_CSWITCH_CUM

THREAD_FORKPROC_FORK

THREAD_FORK_CUMPROC_FORK_CUM

THREAD_GRAPHICS_WAIT_PCTPROC_GRAPHICS_WAIT_PCT

THREAD_GRAPHICS_WAIT_PCT_CUMPROC_GRAPHICS_WAIT_PCT_CUM

THREAD_GRAPHICS_WAIT_TIMEPROC_GRAPHICS_WAIT_TIME

THREAD_GRAPHICS_WAIT_TIME_CUMPROC_GRAPHICS_WAIT_TIME_CUM

THREAD_GROUP_IDPROC_GROUP_ID

THREAD_GROUP_NAMEPROC_GROUP_NAME

THREAD_INODE_WAIT_PCTPROC_INODE_WAIT_PCT

THREAD_INODE_WAIT_PCT_CUMPROC_INODE_WAIT_PCT_CUM

THREAD_INODE_WAIT_TIMEPROC_INODE_WAIT_TIME

THREAD_INODE_WAIT_TIME_CUMPROC_INODE_WAIT_TIME_CUM

THREAD_INTERESTPROC_INTEREST

THREAD_INTERRUPTSPROC_INTERRUPTS

THREAD_INTERRUPTS_CUMPROC_INTERRUPTS_CUM

THREAD_INTERVALPROC_INTERVAL

THREAD_INTERVAL_ALIVEPROC_INTERVAL_ALIVE

THREAD_INTERVAL_CUMPROC_INTERVAL_CUM

THREAD_IOCTLPROC_IOCTL

THREAD_IOCTL_CUMPROC_IOCTL_CUM

THREAD_IO_BYTEPROC_IO_BYTE

THREAD_IO_BYTE_CUMPROC_IO_BYTE_CUM

THREAD_IO_BYTE_RATEPROC_IO_BYTE_RATE

THREAD_IO_BYTE_RATE_CUMPROC_IO_BYTE_RATE_CUM

THREAD_IPC_SUBSYSTEM_WAIT_PCTPROC_IPC_SUBSYSTEM_WAIT_PCT

THREAD_IPC_SUBSYSTEM_WAIT_PCT_CUMPROC_IPC_SUBSYSTEM_WAIT_PCT_CUM

THREAD_IPC_SUBSYSTEM_WAIT_TIMEPROC_IPC_SUBSYSTEM_WAIT_TIME

THREAD_IPC_SUBSYSTEM_WAIT_TIME_CUMPROC_IPC_SUBSYSTEM_WAIT_TIME_
CUM

THREAD_IPC_WAIT_PCTPROC_IPC_WAIT_PCT

THREAD_IPC_WAIT_PCT_CUMPROC_IPC_WAIT_PCT_CUM

THREAD_IPC_WAIT_TIMEPROC_IPC_WAIT_TIME

THREAD_IPC_WAIT_TIME_CUMPROC_IPC_WAIT_TIME_CUM

THREAD_JOBCTL_WAIT_PCTPROC_JOBCTL_WAIT_PCT

THREAD_JOBCTL_WAIT_PCT_CUMPROC_JOBCTL_WAIT_PCT_CUM

THREAD_JOBCTL_WAIT_TIMEPROC_JOBCTL_WAIT_TIME

THREAD_JOBCTL_WAIT_TIME_CUMPROC_JOBCTL_WAIT_TIME_CUM

THREAD_LAN_WAIT_PCTPROC_LAN_WAIT_PCT

THREAD_LAN_WAIT_PCT_CUMPROC_LAN_WAIT_PCT_CUM

THREAD_LAN_WAIT_TIMEPROC_LAN_WAIT_TIME

THREAD_LAN_WAIT_TIME_CUMPROC_LAN_WAIT_TIME_CUM

THREAD_MAJOR_FAULTPROC_MAJOR_FAULT

THREAD_MAJOR_FAULT_CUMPROC_MAJOR_FAULT_CUM

THREAD_MEM_PRIVATE_RESPROC_MEM_PRIVATE_RES

THREAD_MEM_RESPROC_MEM_RES

THREAD_MEM_RES_HIGHPROC_MEM_RES_HIGH

THREAD_MEM_SHARED_RESPROC_MEM_SHARED_RES

THREAD_MEM_VFAULT_COUNTPROC_MEM_VFAULT_COUNT

THREAD_MEM_VFAULT_COUNT_CUMPROC_MEM_VFAULT_COUNT_CUM

THREAD_MEM_VIRTPROC_MEM_VIRT

THREAD_MEM_WAIT_PCTPROC_MEM_WAIT_PCT

THREAD_MEM_WAIT_PCT_CUMPROC_MEM_WAIT_PCT_CUM

THREAD_MEM_WAIT_TIMEPROC_MEM_WAIT_TIME

THREAD_MEM_WAIT_TIME_CUMPROC_MEM_WAIT_TIME_CUM

THREAD_MINOR_FAULTPROC_MINOR_FAULT

THREAD_MINOR_FAULT_CUMPROC_MINOR_FAULT_CUM

THREAD_MSG_RECEIVEDPROC_MSG_RECEIVED

THREAD_MSG_RECEIVED_CUMPROC_MSG_RECEIVED_CUM

THREAD_MSG_SENTPROC_MSG_SENT

THREAD_MSG_SENT_CUMPROC_MSG_SENT_CUM

THREAD_MSG_WAIT_PCTPROC_MSG_WAIT_PCT

THREAD_MSG_WAIT_PCT_CUMPROC_MSG_WAIT_PCT_CUM

THREAD_MSG_WAIT_TIMEPROC_MSG_WAIT_TIME

THREAD_MSG_WAIT_TIME_CUMPROC_MSG_WAIT_TIME_CUM

THREAD_NFS_WAIT_PCTPROC_NFS_WAIT_PCT

THREAD_NFS_WAIT_PCT_CUMPROC_NFS_WAIT_PCT_CUM

THREAD_NFS_WAIT_TIMEPROC_NFS_WAIT_TIME

THREAD_NFS_WAIT_TIME_CUMPROC_NFS_WAIT_TIME_CUM

THREAD_NICE_PRIPROC_NICE_PRI

THREAD_NONDISK_LOGL_READPROC_NONDISK_LOGL_READ

THREAD_NONDISK_LOGL_READ_CUMPROC_NONDISK_LOGL_READ_CUM

THREAD_NONDISK_LOGL_WRITEPROC_NONDISK_LOGL_WRITE

THREAD_NONDISK_LOGL_WRITE_CUMPROC_NONDISK_LOGL_WRITE_CUM

THREAD_NONDISK_PHYS_READPROC_NONDISK_PHYS_READ

THREAD_NONDISK_PHYS_READ_CUMPROC_NONDISK_PHYS_READ_CUM

THREAD_NONDISK_PHYS_WRITEPROC_NONDISK_PHYS_WRITE

THREAD_NONDISK_PHYS_WRITE_CUMPROC_NONDISK_PHYS_WRITE_CUM

THREAD_OPENPROC_OPEN

THREAD_OPEN_CUMPROC_OPEN_CUM

THREAD_OTHER_IO_WAIT_PCTPROC_OTHER_IO_WAIT_PCT

THREAD_OTHER_IO_WAIT_PCT_CUMPROC_OTHER_IO_WAIT_PCT_CUM

THREAD_OTHER_IO_WAIT_TIMEPROC_OTHER_IO_WAIT_TIME

THREAD_OTHER_IO_WAIT_TIME_CUMPROC_OTHER_IO_WAIT_TIME_CUM

THREAD_OTHER_WAIT_PCTPROC_OTHER_WAIT_PCT

THREAD_OTHER_WAIT_PCT_CUMPROC_OTHER_WAIT_PCT_CUM

THREAD_OTHER_WAIT_TIMEPROC_OTHER_WAIT_TIME

THREAD_OTHER_WAIT_TIME_CUMPROC_OTHER_WAIT_TIME_CUM

THREAD_PAGEFAULTPROC_PAGEFAULT

THREAD_PAGEFAULT_RATEPROC_PAGEFAULT_RATE

THREAD_PAGEFAULT_RATE_CUMPROC_PAGEFAULT_RATE_CUM

THREAD_PARENT_PROC_IDPROC_PARENT_PROC_ID

THREAD_PIPE_WAIT_PCTPROC_PIPE_WAIT_PCT

THREAD_PIPE_WAIT_PCT_CUMPROC_PIPE_WAIT_PCT_CUM

THREAD_PIPE_WAIT_TIMEPROC_PIPE_WAIT_TIME

THREAD_PIPE_WAIT_TIME_CUMPROC_PIPE_WAIT_TIME_CUM

THREAD_PRIPROC_PRI

THREAD_PRI_WAIT_PCTPROC_PRI_WAIT_PCT

THREAD_PRI_WAIT_PCT_CUMPROC_PRI_WAIT_PCT_CUM

THREAD_PRI_WAIT_TIMEPROC_PRI_WAIT_TIME

THREAD_PRI_WAIT_TIME_CUMPROC_PRI_WAIT_TIME_CUM

THREAD_PRMIDPROC_PRMID

THREAD_PROC_ARGV1PROC_PROC_ARGV1

THREAD_PROC_CMDPROC_PROC_CMD

THREAD_PROC_IDPROC_PROC_ID

THREAD_PROC_NAMEPROC_PROC_NAME

THREAD_RPC_WAIT_PCTPROC_RPC_WAIT_PCT

THREAD_RPC_WAIT_PCT_CUMPROC_RPC_WAIT_PCT_CUM

THREAD_RPC_WAIT_TIMEPROC_RPC_WAIT_TIME

THREAD_RPC_WAIT_TIME_CUMPROC_RPC_WAIT_TIME_CUM

THREAD_RUN_TIMEPROC_RUN_TIME

THREAD_SCHEDULERPROC_SCHEDULER

THREAD_SEM_WAIT_PCTPROC_SEM_WAIT_PCT

THREAD_SEM_WAIT_PCT_CUMPROC_SEM_WAIT_PCT_CUM

THREAD_SEM_WAIT_TIMEPROC_SEM_WAIT_TIME

THREAD_SEM_WAIT_TIME_CUMPROC_SEM_WAIT_TIME_CUM

THREAD_SIGNALPROC_SIGNAL

THREAD_SIGNAL_CUMPROC_SIGNAL_CUM

THREAD_SLEEP_WAIT_PCTPROC_SLEEP_WAIT_PCT

THREAD_SLEEP_WAIT_PCT_CUMPROC_SLEEP_WAIT_PCT_CUM

THREAD_SLEEP_WAIT_TIMEPROC_SLEEP_WAIT_TIME

THREAD_SLEEP_WAIT_TIME_CUMPROC_SLEEP_WAIT_TIME_CUM

THREAD_SOCKET_WAIT_PCTPROC_SOCKET_WAIT_PCT

THREAD_SOCKET_WAIT_PCT_CUMPROC_SOCKET_WAIT_PCT_CUM

THREAD_SOCKET_WAIT_TIMEPROC_SOCKET_WAIT_TIME

THREAD_SOCKET_WAIT_TIME_CUMPROC_SOCKET_WAIT_TIME_CUM

THREAD_STARTTIMEPROC_STARTTIME

THREAD_STATEPROC_STATE

THREAD_STOP_REASONPROC_STOP_REASON

THREAD_STOP_REASON_FLAGPROC_STOP_REASON_FLAG

THREAD_STREAM_WAIT_PCTPROC_STREAM_WAIT_PCT

THREAD_STREAM_WAIT_PCT_CUMPROC_STREAM_WAIT_PCT_CUM

THREAD_STREAM_WAIT_TIMEPROC_STREAM_WAIT_TIME

THREAD_STREAM_WAIT_TIME_CUMPROC_STREAM_WAIT_TIME_CUM

THREAD_SWAPPROC_SWAP

THREAD_SWAP_CUMPROC_SWAP_CUM

THREAD_SYS_WAIT_PCTPROC_SYS_WAIT_PCT

THREAD_SYS_WAIT_PCT_CUMPROC_SYS_WAIT_PCT_CUM

THREAD_SYS_WAIT_TIMEPROC_SYS_WAIT_TIME

THREAD_SYS_WAIT_TIME_CUMPROC_SYS_WAIT_TIME_CUM

THREAD_TERM_IO_WAIT_PCTPROC_TERM_IO_WAIT_PCT

THREAD_TERM_IO_WAIT_PCT_CUMPROC_TERM_IO_WAIT_PCT_CUM

THREAD_TERM_IO_WAIT_TIMEPROC_TERM_IO_WAIT_TIME

THREAD_TERM_IO_WAIT_TIME_CUMPROC_TERM_IO_WAIT_TIME_CUM

THREAD_THREAD_COUNTPROC_THREAD_COUNT

THREAD_THREAD_IDPROC_THREAD_ID

THREAD_TIMEPROC_TIME

THREAD_TOP_CPU_INDEXPROC_TOP_CPU_INDEX

THREAD_TOP_DISK_INDEXPROC_TOP_DISK_INDEX

THREAD_TOTAL_WAIT_TIMEPROC_TOTAL_WAIT_TIME

THREAD_TOTAL_WAIT_TIME_CUMPROC_TOTAL_WAIT_TIME_CUM

THREAD_TTYPROC_TTY

THREAD_TTY_DEVPROC_TTY_DEV

THREAD_UIDPROC_UID

THREAD_USER_NAMEPROC_USER_NAME

THREAD_USER_THREAD_IDPROC_USER_THREAD_ID

THREAD_USRPRIPROC_USRPRI

THREAD_VOLUNTARY_CSWITCHPROC_VOLUNTARY_CSWITCH

THREAD_VOLUNTARY_CSWITCH_CUMPROC_VOLUNTARY_CSWITCH_CUM

Network by Logical Detail Metrics

BYNETIF_LOGL_INTERVALBYNETIF_LOGL_INTERVAL

BYNETIF_LOGL_INTERVAL_CUMBYNETIF_LOGL_INTERVAL_CUM

BYNETIF_LOGL_IN_PACKETBYNETIF_LOGL_IN_PACKET

BYNETIF_LOGL_IN_PACKET_RATEBYNETIF_LOGL_IN_PACKET_RATE

BYNETIF_LOGL_IN_PACKET_RATE_CUMBYNETIF_LOGL_IN_PACKET_RATE_CUM

BYNETIF_LOGL_IP_ADDRESSBYNETIF_LOGL_IP_ADDRESS

BYNETIF_LOGL_NAMEBYNETIF_LOGL_NAME

BYNETIF_LOGL_OUT_PACKETBYNETIF_LOGL_OUT_PACKET

BYNETIF_LOGL_OUT_PACKET_RATEBYNETIF_LOGL_OUT_PACKET_RATE

BYNETIF_LOGL_OUT_PACKET_RATE_CUMBYNETIF_LOGL_OUT_PACKET_RATE_CUM

Transaction Client Metrics

TT_CLIENT_ABORTTT_ABORT

TT_CLIENT_ABORT_CUMTT_ABORT_CUM

TT_CLIENT_ABORT_WALL_TIMETT_ABORT_WALL_TIME

TT_CLIENT_ABORT_WALL_TIME_CUMTT_ABORT_WALL_TIME_CUM

TT_CLIENT_ADDRESSTT_CLIENT_ADDRESS

TT_CLIENT_ADDRESS_FORMATTT_CLIENT_ADDRESS_FORMAT

TT_CLIENT_TRAN_IDTT_CLIENT_TRAN_ID

TT_CLIENT_COUNTTT_COUNT

TT_CLIENT_COUNT_CUMTT_COUNT_CUM

TT_CLIENT_FAILEDTT_FAILED

TT_CLIENT_FAILED_CUMTT_FAILED_CUM

TT_CLIENT_FAILED_WALL_TIMETT_FAILED_WALL_TIME

TT_CLIENT_FAILED_WALL_TIME_CUMTT_FAILED_WALL_TIME_CUM

TT_CLIENT_INTERVALTT_INTERVAL

TT_CLIENT_INTERVAL_CUMTT_INTERVAL_CUM

TT_CLIENT_SLO_COUNTTT_SLO_COUNT

TT_CLIENT_SLO_COUNT_CUMTT_SLO_COUNT_CUM

TT_CLIENT_UPDATETT_UPDATE

TT_CLIENT_UPDATE_CUMTT_UPDATE_CUM

TT_CLIENT_WALL_TIMETT_WALL_TIME

TT_CLIENT_WALL_TIME_CUMTT_WALL_TIME_CUM

TT_CLIENT_WALL_TIME_PER_TRANTT_WALL_TIME_PER_TRAN

TT_CLIENT_WALL_TIME_PER_TRAN_CUMTT_WALL_TIME_PER_TRAN_CUM

Transaction Instance Metrics

TT_INSTANCE_IDTT_INSTANCE_ID

TT_INSTANCE_PROC_IDTT_INSTANCE_PROC_ID

TT_INSTANCE_START_TIMETT_INSTANCE_START_TIME

TT_INSTANCE_STOP_TIMETT_INSTANCE_STOP_TIME

TT_INSTANCE_THREAD_IDTT_INSTANCE_THREAD_ID

TT_INSTANCE_UPDATE_COUNTTT_INSTANCE_UPDATE_COUNT

TT_INSTANCE_UPDATE_TIMETT_INSTANCE_UPDATE_TIME

TT_INSTANCE_WALL_TIMETT_INSTANCE_WALL_TIME

Transaction User Defined Measurement Metrics

TT_USER_MEASUREMENT_AVGTT_USER_MEASUREMENT_AVG

TT_USER_MEASUREMENT_COUNTTT_USER_MEASUREMENT_COUNT

TT_USER_MEASUREMENT_MAXTT_USER_MEASUREMENT_MAX

TT_USER_MEASUREMENT_MINTT_USER_MEASUREMENT_MIN

TT_USER_MEASUREMENT_NAMETT_USER_MEASUREMENT_NAME

TT_USER_MEASUREMENT_STRING1024_VALUETT_USER_MEASUREMENT_STRING1024_VALUE

TT_USER_MEASUREMENT_STRING32_VALUETT_USER_MEASUREMENT_STRING32_VALUE

TT_USER_MEASUREMENT_TYPETT_USER_MEASUREMENT_TYPE

TT_USER_MEASUREMENT_VALUETT_USER_MEASUREMENT_VALUE

Transaction Client User Defined Measurement Metrics

TT_CLIENT_USER_MEASUREMENT_AVGTT_USER_MEASUREMENT_AVG

TT_CLIENT_USER_MEASUREMENT_COUNTTT_USER_MEASUREMENT_COUNT

TT_CLIENT_USER_MEASUREMENT_MAXTT_USER_MEASUREMENT_MAX

TT_CLIENT_USER_MEASUREMENT_MINTT_USER_MEASUREMENT_MIN

TT_CLIENT_USER_MEASUREMENT_NAMETT_USER_MEASUREMENT_NAME

TT_CLIENT_USER_MEASUREMENT_STRING1024_VALUETT_USER_MEASUREMENT_STRING1024_VALUE

TT_CLIENT_USER_MEASUREMENT_STRING32_VALUETT_USER_MEASUREMENT_STRING32_VALUE

TT_CLIENT_USER_MEASUREMENT_TYPETT_USER_MEASUREMENT_TYPE

TT_CLIENT_USER_MEASUREMENT_VALUETT_USER_MEASUREMENT_VALUE

Transaction Instance User Defined Measurement Metrics

TT_INSTANCE_USER_MEASUREMENT_AVGTT_USER_MEASUREMENT_AVG

TT_INSTANCE_USER_MEASUREMENT_COUNTTT_USER_MEASUREMENT_COUNT

TT_INSTANCE_USER_MEASUREMENT_MAXTT_USER_MEASUREMENT_MAX

TT_INSTANCE_USER_MEASUREMENT_MINTT_USER_MEASUREMENT_MIN

TT_INSTANCE_USER_MEASUREMENT_NAMETT_USER_MEASUREMENT_NAME

TT_INSTANCE_USER_MEASUREMENT_STRING1024_VALUETT_USER_MEASUREMENT_STRING1024_VALUE

TT_INSTANCE_USER_MEASUREMENT_STRING32_VALUETT_USER_MEASUREMENT_STRING32_VALUE

TT_INSTANCE_USER_MEASUREMENT_TYPETT_USER_MEASUREMENT_TYPE

TT_INSTANCE_USER_MEASUREMENT_VALUETT_USER_MEASUREMENT_VALUE

PRM By Volume Group Metrics

PRM_BYVG_GROUP_ENTITLEMENTPRM_BYVG_GROUP_ENTITLEMENT

PRM_BYVG_GROUP_UTILPRM_BYVG_GROUP_UTIL

PRM_BYVG_INTERVALPRM_BYVG_INTERVAL

PRM_BYVG_INTERVAL_CUMPRM_BYVG_INTERVAL_CUM

PRM_BYVG_PRM_GROUPIDPRM_BYVG_PRM_GROUPID

PRM_BYVG_PRM_GROUPNAMEPRM_BYVG_PRM_GROUPNAME

PRM_BYVG_REQUESTPRM_BYVG_REQUEST

PRM_BYVG_REQUEST_CUMPRM_BYVG_REQUEST_CUM

PRM_BYVG_REQUEST_QUEUEPRM_BYVG_REQUEST_QUEUE

PRM_BYVG_TRANSFERPRM_BYVG_TRANSFER

PRM_BYVG_TRANSFER_CUMPRM_BYVG_TRANSFER_CUM

By Logical System Metrics

BYLS_CPU_CYCLE_ENTL_MAXBYLS_CPU_CYCLE_ENTL_MAX

BYLS_CPU_CYCLE_ENTL_MINBYLS_CPU_CYCLE_ENTL_MIN

BYLS_CPU_ENTL_MAXBYLS_CPU_ENTL_MAX

BYLS_CPU_ENTL_MINBYLS_CPU_ENTL_MIN

BYLS_CPU_ENTL_UTILBYLS_CPU_ENTL_UTIL

BYLS_CPU_PHYSCBYLS_CPU_PHYSC

BYLS_CPU_PHYS_TOTAL_TIMEBYLS_CPU_PHYS_TOTAL_TIME

BYLS_CPU_PHYS_TOTAL_TIME_CUMBYLS_CPU_PHYS_TOTAL_TIME_CUM

BYLS_CPU_PHYS_TOTAL_UTILBYLS_CPU_PHYS_TOTAL_UTIL

BYLS_CPU_TOTAL_UTILBYLS_CPU_TOTAL_UTIL

BYLS_DISPLAY_NAMEBYLS_DISPLAY_NAME

BYLS_IP_ADDRESSBYLS_IP_ADDRESS

BYLS_LS_HOSTNAMEBYLS_LS_HOSTNAME

BYLS_LS_IDBYLS_LS_ID

BYLS_LS_MODEBYLS_LS_MODE

BYLS_LS_NAMEBYLS_LS_NAME

BYLS_LS_OSTYPEBYLS_LS_OSTYPE

BYLS_LS_PROC_IDBYLS_LS_PROC_ID

BYLS_LS_SHAREDBYLS_LS_SHARED

BYLS_LS_STATEBYLS_LS_STATE

BYLS_LS_UUIDBYLS_LS_UUID

BYLS_MEM_ENTLBYLS_MEM_ENTL

BYLS_MEM_ENTL_MAXBYLS_MEM_ENTL_MAX

BYLS_MEM_ENTL_MINBYLS_MEM_ENTL_MIN

BYLS_MEM_ENTL_UTILBYLS_MEM_ENTL_UTIL

BYLS_MEM_FREEBYLS_MEM_FREE

BYLS_MEM_FREE_UTILBYLS_MEM_FREE_UTIL

BYLS_MEM_HEALTHBYLS_MEM_HEALTH

BYLS_MEM_PHYSBYLS_MEM_PHYS

BYLS_MEM_PHYS_UTILBYLS_MEM_PHYS_UTIL

BYLS_MEM_USEDBYLS_MEM_USED

BYLS_NUM_CPUBYLS_NUM_CPU

BYLS_NUM_DISKBYLS_NUM_DISK

BYLS_NUM_NETIFBYLS_NUM_NETIF

BYLS_UPTIME_SECONDSBYLS_UPTIME_SECONDS

By Hba Metrics

BYHBA_AVG_SERVICE_TIMEBYHBA_AVG_SERVICE_TIME

BYHBA_AVG_WAIT_TIMEBYHBA_AVG_WAIT_TIME

BYHBA_BUSY_TIMEBYHBA_BUSY_TIME

BYHBA_BYTE_RATEBYHBA_BYTE_RATE

BYHBA_BYTE_RATE_CUMBYHBA_BYTE_RATE_CUM

BYHBA_CLASSBYHBA_CLASS

BYHBA_DEVNAMEBYHBA_DEVNAME

BYHBA_DEVNOBYHBA_DEVNO

BYHBA_DRIVERBYHBA_DRIVER

BYHBA_IDBYHBA_ID

BYHBA_INTERVALBYHBA_INTERVAL

BYHBA_INTERVAL_CUMBYHBA_INTERVAL_CUM

BYHBA_IOBYHBA_IO

BYHBA_IO_RATEBYHBA_IO_RATE

BYHBA_IO_RATE_CUMBYHBA_IO_RATE_CUM

BYHBA_NAMEBYHBA_NAME

BYHBA_READBYHBA_READ

BYHBA_READ_BYTE_RATEBYHBA_READ_BYTE_RATE

BYHBA_READ_BYTE_RATE_CUMBYHBA_READ_BYTE_RATE_CUM

BYHBA_READ_RATEBYHBA_READ_RATE

BYHBA_READ_RATE_CUMBYHBA_READ_RATE_CUM

BYHBA_REQUEST_QUEUEBYHBA_REQUEST_QUEUE

BYHBA_STATEBYHBA_STATE

BYHBA_THROUGHPUT_UTILBYHBA_THROUGHPUT_UTIL

BYHBA_TIMEBYHBA_TIME

BYHBA_TYPEBYHBA_TYPE

BYHBA_UTILBYHBA_UTIL

BYHBA_WRITEBYHBA_WRITE

BYHBA_WRITE_BYTE_RATEBYHBA_WRITE_BYTE_RATE

BYHBA_WRITE_BYTE_RATE_CUMBYHBA_WRITE_BYTE_RATE_CUM

BYHBA_WRITE_RATEBYHBA_WRITE_RATE

BYHBA_WRITE_RATE_CUMBYHBA_WRITE_RATE_CUM

Metric Definitions

APP_ACTIVE_APP

The number of applications that had processes active (consuming cpu resources) during the interval.

APP_ACTIVE_APP_PRM

The number of PRM groups with at least one process that had activity during the interval.

APP_ACTIVE_PROC

An active process is one that exists and consumes some CPU time. APP_ACTIVE_PROC is the sum of the alive-process-time/interval-time ratios of every process belonging to an application that is active (uses any CPU time) during an interval.

The following diagram of a four second interval showing two processes, A and B, for an application should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

```
     ----------- Seconds -----------
       1          2          3      4
Proc
---- ----       ----       ----   ----
A    live       live       live   live

B    live/CPU   live/CPU   live   dead
```

Process A is alive for the entire four second interval, but consumes no CPU. A's contribution to APP_ALIVE_PROC is 4*1/4. A contributes 0*1/4 to APP_ACTIVE_PROC. B's contribution to APP_ALIVE_PROC is 3*1/4. B contributes 2*1/4 to APP_ACTIVE_PROC. Thus, for this interval, APP_ACTIVE_PROC equals 0.5 and APP_ALIVE_PROC equals 1.75.

Because a process may be alive but not active, APP_ACTIVE_PROC will always be less than or equal to APP_ALIVE_PROC.

This metric indicates the number of processes in an application group that are competing for the CPU. This metric is useful, along with other metrics, for comparing loads placed on the system by different groups of processes.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_ALIVE_PROC

An alive process is one that exists on the system. APP_ALIVE_PROC is the sum of the alive-process-time/interval-time ratios for every process belonging to a given application.

The following diagram of a four second interval showing two processes, A and B, for an application should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

```
      ----------- Seconds -----------
        1          2          3       4
Proc
---- ----       ----       ----    ----
A    live       live       live    live

B    live/CPU   live/CPU   live    dead
```

Process A is alive for the entire four second interval but consumes no CPU. A's contribution to APP_ALIVE_PROC is 4*1/4. A contributes 0*1/4 to APP_ACTIVE_PROC. B's contribution to APP_ALIVE_PROC is 3*1/4. B contributes 2*1/4 to APP_ACTIVE_PROC. Thus, for this interval, APP_ACTIVE_PROC equals 0.5 and APP_ALIVE_PROC equals 1.75.

Because a process may be alive but not active, APP_ACTIVE_PROC will always be less than or equal to APP_ALIVE_PROC.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_COMPLETED_PROC

The number of processes in this group that completed during the interval.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_CPU_NICE_TIME

The time, in seconds, that processes in this group were using the CPU in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_NICE_UTIL

The percentage of time that processes in this group were using the CPU in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_NNICE_TIME

The time, in seconds, that processes in this group were using the CPU in user mode at a nice priority calculated from using negative nice values during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_NNICE_UTIL

The percentage of time that processes in this group were using the CPU in user mode at a nice priority calculated from using negative nice values during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup.

Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_NORMAL_TIME

The time, in seconds, that processes in this group were in user mode at a normal priority during the interval.

Normal priority user mode CPU excludes CPU used at real-time and nice priorities. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_NORMAL_UTIL

The percentage of time that processes in this group were in user mode running at normal priority during the interval. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_REALTIME_TIME

The time, in seconds, that the processes in this group were in user mode at a “realtime” priority during the interval. “Realtime” priority is 0-127. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_REALTIME_UTIL

The percentage of time that processes in this group were in user mode at a “realtime” priority during the interval. “Realtime” priority is 0-127.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_SYS_MODE_TIME

The time, in seconds, during the interval that the CPU was in system mode for processes in this group.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_SYS_MODE_UTIL

The percentage of time during the interval that the CPU was used in system mode for processes in this group.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

High system CPU utilizations are normal for IO intensive groups. Abnormally high system CPU utilization can indicate that a hardware problem is causing a high interrupt rate. It can also indicate programs that are not making efficient system calls. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_TOTAL_TIME

The total CPU time, in seconds, devoted to processes in this group during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_TOTAL_UTIL

The percentage of the total CPU time devoted to processes in this group during the interval. This indicates the relative CPU load placed on the system by processes in this group.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

Large values for this metric may indicate that this group is causing a CPU bottleneck. This would be normal in a computation-bound workload, but might mean that processes are using excessive CPU time and perhaps looping.

If the “other” application shows significant amounts of CPU, you may want to consider tuning your parm file so that process activity is accounted for in known applications.

```
  APP_CPU_TOTAL_UTIL =
    APP_CPU_SYS_MODE_UTIL +
    APP_CPU_USER_MODE_UTIL
```

NOTE: On Windows, the sum of the APP_CPU_TOTAL_UTIL metrics may not equal GBL_CPU_TOTAL_UTIL. Microsoft states that “this is expected behavior” because the GBL_CPU_TOTAL_UTIL metric is taken from the NT performance library Processor objects while the APP_CPU_TOTAL_UTIL metrics are taken from the Process objects. Microsoft states that there can be CPU time accounted for in the Processor system objects that may not be seen in the Process objects. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_TOTAL_UTIL_CUM

The average CPU time per interval for processes in this group over the cumulative collection time, or since the last PRM configuration change on HP-UX.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_USER_MODE_TIME

The time, in seconds, that processes in this group were in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_USER_MODE_UTIL

The percentage of time that processes in this group were using the CPU in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

High user mode CPU percentages are normal for computation-intensive groups. Low values of user CPU utilization compared to relatively high values for APP_CPU_SYS_MODE_UTIL can indicate a hardware problem or improperly tuned programs in this group.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total

processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_DISK_FS_IO_RATE

The number of file system disk IOs for processes in this group during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

APP_DISK_LOGL_IO_RATE

The number of logical IOs per second for processes in this group during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

APP_DISK_LOGL_READ

The number of logical reads for processes in this group during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

APP_DISK_LOGL_READ_RATE

The number of logical reads per second for processes in this group during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

APP_DISK_LOGL_WRITE

The number of logical writes for processes in this group during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

APP_DISK_LOGL_WRITE_RATE

The number of logical writes per second for processes in this group during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

APP_DISK_PHYS_IO_RATE

The number of physical IOs per second for processes in this group during the interval.

APP_DISK_PHYS_READ

The number of physical reads for processes in this group during the interval.

APP_DISK_PHYS_READ_RATE

The number of physical reads per second for processes in this group during the interval.

APP_DISK_PHYS_WRITE

The number of physical writes for processes in this group during the interval.

APP_DISK_PHYS_WRITE_RATE

The number of physical writes per second for processes in this group during the interval.

APP_DISK_RAW_IO_RATE

The total number of raw IOs for processes in this group during the interval. Only accesses to local disk devices are counted.

APP_DISK_SUBSYSTEM_QUEUE

The average number of processes or kernel threads in this group that were blocked on the disk subsystem (waiting for their file system IOs to complete) during the interval.

This is the sum of processes or kernel threads in the DISK, INODE, CACHE and CDFS wait states. It does not include processes or kernel threads doing raw IO to disk devices.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_DISK_SUBSYSTEM_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on the disk subsystem (waiting for their file system IOs to complete) during the interval.

This is the sum of processes or kernel threads in the DISK, INODE, CACHE and CDFS wait states. It does not include processes or kernel threads doing raw IO to disk devices.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_DISK_SYSTEM_IO_RATE

The number of physical IOs per second generated by the kernel for file system management (inode accesses or updates) for processes in this group during the interval.

APP_DISK_VM_IO_RATE

The number of virtual memory IOs per second made on behalf of processes in this group during the interval.

IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

APP_INTERVAL

The amount of time in the interval.

APP_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

APP_IO_BYTE

The number of characters (in KB) transferred for processes in this group to all devices during the interval. This includes IO to disk, terminal, tape and printers.

APP_IO_BYTE_RATE

The number of characters (in KB) per second transferred for processes in this group to all devices during the interval. This includes IO to disk, terminal, tape and printers.

APP_IPC_SUBSYSTEM_QUEUE

The average number of processes or kernel threads in this group blocked on the InterProcess Communication (IPC) subsystems (waiting for their interprocess communication activity to complete) during the interval.

This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on (IPC + MSG + SEM + PIPE + SOCKT + STRMS) divided by the interval time.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_IPC_SUBSYSTEM_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on the InterProcess Communication (IPC) subsystems (waiting for their interprocess communication activity to complete) during the interval.

This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_MAJOR_FAULT

The number of major page faults that required a disk IO for processes in this group during the interval.

APP_MAJOR_FAULT_RATE

The number of major page faults per second that required a disk IO for processes in this group during the interval.

APP_MEM_QUEUE

The average number of processes or kernel threads in this group blocked on memory (waiting for virtual memory disk accesses to complete) during the interval.

This typically happens when processes or kernel threads are allocating a large amount of memory. It can also happen when processes or kernel threads access memory that has been paged out to disk (deactivated) because of overall memory pressure on the system. Note that large programs can block on VM disk access when they are initializing, bringing their text and data pages into memory.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_MEM_RES

On Unix systems, this is the sum of the size (in MB) of resident memory for processes in this group that were alive at the end of the interval. This consists of text, data, stack, and shared memory regions.

On HP-UX, since PROC_MEM_RES typically takes shared region references into account, this approximates the total resident (physical) memory consumed by all processes in this group.

On all other Unix systems, this is the sum of the resident memory region sizes for all processes in this group. When the resident memory size for processes includes shared regions, such as shared memory and library text and data, the shared regions are counted multiple times in this sum. For example, if the application contains four processes that are attached to a 500MB shared memory region that is all resident in physical memory, then 2000MB is contributed towards the sum in this metric. As such, this metric can overestimate the resident memory being used by processes in this group when they share memory regions.

Refer to the help text for PROC_MEM_RES for additional information.

On Windows, this is the sum of the size (in MB) of the working sets for processes in this group during the interval. The working set counts memory pages referenced recently by the threads making up this group. Note that the size of the working set is often larger than the amount of pagefile space consumed.

APP_MEM_UTIL

On Unix systems, this is the approximate percentage of the system's physical memory used as resident memory by processes in this group that were alive at the end of the interval. This metric summarizes process private and shared memory in each application.

On Windows, this is an estimate of the percentage of the system's physical memory allocated for working set memory by processes in this group during the interval.

On HP-UX, this consists of text, data, stack, as well the process' portion of shared memory regions (such as, shared libraries, text segments, and shared data). The sum of the shared region pages is typically divided by the number of references.

APP_MEM_VIRT

On Unix systems, this is the sum (in MB) of virtual memory for processes in this group that were alive at the end of the interval. This consists of text, data, stack, and shared memory regions.

On HP-UX, since PROC_MEM_VIRT typically takes shared region references into account, this approximates the total virtual memory consumed by all processes in this group.

On all other Unix systems, this is the sum of the virtual memory region sizes for all processes in this group. When the virtual memory size for processes includes shared regions, such as shared memory and library text and data, the shared regions are counted multiple times in this sum. For example, if the application contains four processes that are attached to a 500MB shared memory region, then 2000MB is reported in this metric. As such, this metric can overestimate the virtual memory being used by processes in this group when they share memory regions.

On Windows, this is the sum (in MB) of paging file space used for all processes in this group during the interval. Groups of processes may have working set sizes (APP_MEM_RES) larger than the size of their pagefile space.

APP_MEM_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on memory (waiting for virtual memory disk accesses to complete) during the interval.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_MINOR_FAULT

The number of minor page faults satisfied in memory (a page was reclaimed from one of the free lists) for processes in this group during the interval.

APP_MINOR_FAULT_RATE

The number of minor page faults per second satisfied in memory (pages were reclaimed from one of the free lists) for processes in this group during the interval.

APP_NAME

The name of the application (up to 20 characters). This comes from the parm file where the applications are defined.

The application called “other” captures all processes not aggregated into applications specifically defined in the parm file. In other words, if no applications are defined in the parm file, then all process data would be reflected in the “other” application.

APP_NAME_PRM_GROUPNAME

The PRM group name. The PRM group configuration is kept in the PRM configuration file.

APP_NETWORK_SUBSYSTEM_QUEUE

The average number of processes or kernel threads in this group were blocked on the network subsystem (waiting for their network activity to complete) during the interval.

This is the sum of processes or kernel threads in the LAN, NFS, and RPC wait states. This does not include processes or kernel threads blocked on SOCKT (that is, socket) waits, as some processes or kernel threads sit idle in SOCKT waits for long periods.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_NETWORK_SUBSYSTEM_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on the network subsystem (waiting for their network activity to complete) during the interval.

This is the sum of processes or kernel threads in the LAN, NFS, and RPC wait states. This does not include processes or kernel threads blocked on SOCKT (that is, socket) waits, as some processes or kernel threads sit idle in SOCKT waits for long periods.

This is calculated as the accumulated time that all processes or kernel threads in this group spent blocked on (LAN + NFS + RPC) divided by the interval time.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_NUM

The sequentially assigned number of this application or, on Solaris, the project ID when application grouping by project is enabled.

APP_OTHER_IO_QUEUE

The average number of processes or kernel threads in this group that were blocked on “other IO” during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

This is calculated as the accumulated time that all processes or kernel threads in this group spent blocked on other IO divided by the interval time.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_OTHER_IO_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on “other IO” during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_PRI

On Unix systems, this is the average priority of the processes in this group during the interval.

On Windows, this is the average base priority of the processes in this group during the interval.

APP_PRI_QUEUE

The average number of processes or kernel threads in this group blocked on PRI (waiting for their priority to become high enough to get the CPU) during the interval.

This is calculated as the accumulated time that all processes or kernel threads in this group spent blocked on PRI divided by the interval time.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_PRI_STD_DEV

The standard deviation of priorities of the processes in this group during the interval.

This metric is available on HP-UX 10.20.

APP_PRI_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on PRI (waiting for their priority to become high enough to get the CPU) during the interval.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_PRM_CPUCAP_MODE

The PRM CPU Cap Mode state on this system:

```
 0 = PRM is not installed or not
     configured.
 1 = CPU Cap Mode is not enabled
     (PRM CPU entitlements are
     in effect)
 2 = CPU Cap Mode is enabled
     (The PRM CPU entitlements
     behave as caps or limits)
```

APP_PRM_CPU_ENTITLEMENT

The PRM CPU entitlement for this PRM Group ID entry as defined in the PRM configuration file.

APP_PRM_CPU_TOTAL_UTIL_CUM

The average CPU time per interval for processes in this group over the cumulative collection time, or since the last PRM configuration change.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_PRM_DISK_STATE

The PRM DISK state on this system:

```
0 = PRM is not installed or no
    disk specification
1 = reset (PRM is installed in
    reset condition or no disk
    specification)
2 = configured/disabled (The PRM
    disk management is configured)
3 = enabled/configured (The PRM
    disk management is enabled and
    volume groups are configured)
4 = enabled/unconfigured (The PRM
    disk management is enabled,
    however, no volume groups are
    configured)
```

APP_PRM_GROUPID

The PRM Group ID. The PRM group configuration is kept in the PRM configuration file.

APP_PRM_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

APP_PRM_MEM_AVAIL

PRM available memory is the amount of physical memory less the amount of memory reserved for the kernel and system processes running in the PRM_SYS group 0. PRM available memory is a dynamic value that changes with system usage.

APP_PRM_MEM_ENTITLEMENT

The PRM MEM entitlement for this PRM Group ID entry as defined in the PRM configuration file.

APP_PRM_MEM_STATE

The PRM MEM state on this system:

```
0 = PRM is not installed or no
    memory specification
1 = reset (PRM is installed in
    reset condition or no memory
    specification)
2 = configured/disabled (The PRM
    memory scheduler is configured,
    but the standard HP-UX
    scheduler is in effect)
3 = enabled (The PRM memory
    scheduler is configured and
    in effect)
```

APP_PRM_MEM_UPPERBOUND

The PRM MEM upperbound for this PRM Group ID entry as defined in the PRM configuration file.

APP_PRM_MEM_UTIL

The percent of PRM memory used by processes (process private space plus a process' portion of shared memory) within the PRM groups during the interval.

PRM available memory is the amount of physical memory less the amount of memory reserved for the kernel and system processes running in the PRM_SYS group 0. PRM available memory is a dynamic value that changes with system usage.

APP_PRM_STATE

The PRM CPU state on this system:

```
0 = PRM is not installed
1 = reset (PRM is configured with
    only the system group.  The
    standard HP-UX CPU scheduler
    is in effect)
2 = configured/disabled (the PRM
    CPU scheduler is configured,
    but the standard HP-UX
    scheduler is in effect)
3 = enabled (the PRM CPU scheduler
    is configured and in effect)
```

APP_PRM_SUSPENDED_PROC

The number of processes within the PRM groups that were suspended during the interval.

APP_PROC_RUN_TIME

The average run time for processes in this group that completed during the interval.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_SAMPLE

The number of samples of process data that have been averaged or accumulated during this sample.

APP_SEM_QUEUE

The average number of processes or kernel threads in this group that were blocked onsemaphores (waiting for their semaphore operations to complete) during the interval.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_SEM_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked onsemaphores (waiting for their semaphore operations to complete) during the interval.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_SLEEP_QUEUE

The average number of processes or kernel threads in this group that were blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_SLEEP_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

The Application QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues, within the context of a specific application.

The Application WAIT PCT metrics, which are also based on block states, represent the percentage of processes or kernel threads that were alive on the system within the context of a specific application. These values will vary greatly depending on the application.

No direct comparison is reasonable with the Global Queue metrics since they represent the average number of all processes or kernel threads that were alive on the system. As such, the Application WAIT PCT metrics cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

APP_TERM_IO_QUEUE

The average number of processes or kernel threads in this group that were blocked on terminal IO (waiting for their terminal IO to complete) during the interval.

This metric is available on HP-UX 10.20.

APP_TERM_IO_WAIT_PCT

The percentage of time processes or kernel threads in this group were blocked on terminal IO (waiting for terminal IO to complete) during the interval.

A percentage of time spent in a wait state is calculated as the accumulated time kernel threads belonging to processes in this group spent waiting in this state, divided by accumulated alive time of kernel threads belonging to processes in this group during the interval.

For example, assume an application has 20 kernel threads. During the interval, ten kernel threads slept the entire time, while ten kernel threads waited on terminal input. As a result, the application wait percent values would be 50% for SLEEP and 50% for TERM (that is, terminal IO).

This metric is available on HP-UX 10.20.

APP_TIME

The end time of the measurement interval.

BYCPU_ACTIVE

Indicates whether or not this CPU is online. A CPU that is online is considered active.

For HP-UX and certain versions of Linux, the sar(1M) command allows you to check the status of the system CPUs.

For SUN and DEC, the commands psrinfo(1M) and psradm(1M) allow you to check or change the status of the system CPUs.

For AIX, the pstat(1) command allows you to check the status of the system CPUs.

BYCPU_CPU_CLOCK

The clock speed of the CPU in the current slot. The clock speed is in MHz for the selected CPU.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be “na”, some will be based on what is visible at startup time. All information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

On Linux, this value is always rounded up to the next MHz.

BYCPU_CPU_CSWITCH_TIME

The time, in seconds, that this CPU was performing context switches during the interval. On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_CSWITCH_TIME_CUM

The time, in seconds, that this CPU was performing context switches over the cumulative collection time. On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_CSWITCH_UTIL

The percentage of time that this CPU was performing context switches during the interval. On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_CSWITCH_UTIL_CUM

The percentage of time that this CPU was performing context switches over the cumulative collection time. On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_INTERRUPT_TIME

The time, in seconds, that this CPU was performing interrupt processing during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_INTERRUPT_TIME_CUM

The time, in seconds, that this CPU was performing interrupt processing over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_INTERRUPT_UTIL

The percentage of time that this CPU was performing interrupt processing during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_INTERRUPT_UTIL_CUM

The percentage of time that this CPU was performing interrupt processing over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_TIME

The time, in seconds, that this CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_TIME_CUM

The time, in seconds, that this CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_UTIL

The percentage of time that this CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_UTIL_CUM

The average percentage of time that this CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup.

Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NNICE_TIME

The time, in seconds, that this CPU was in user mode at a nice priority calculated from processes with negative nice values during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NNICE_TIME_CUM

The time, in seconds, that this CPU was in user mode at a nice priority calculated from processes with negative nice values over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NNICE_UTIL

The percentage of time that this CPU was in user mode at a nice priority calculated from processes with negative nice values during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NNICE_UTIL_CUM

The average percentage of time that this CPU was in user mode at a nice priority calculated from processes with negative nice values over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NORMAL_TIME

The time, in seconds, that this CPU was running in user mode at a normal priority during the interval. Normal priority user mode CPU excludes CPU used at real-time and nice priorities. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be

added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NORMAL_TIME_CUM

The time, in seconds, that this CPU was running in user mode at a normal priority over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NORMAL_UTIL

The percentage of time that this CPU was running in user mode at a normal priority during the interval. Normal priority user mode CPU excludes CPU used at real-time and nice priorities. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NORMAL_UTIL_CUM

The average percentage of time that this CPU was running in user mode at a normal priority over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_REALTIME_TIME

The time, in seconds, that this CPU was running at a realtime priority during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_REALTIME_TIME_CUM

The time, in seconds, that this CPU was running at a realtime priority over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_REALTIME_UTIL

The percentage of time that this CPU was running at a realtime priority during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_REALTIME_UTIL_CUM

The percentage of time that this CPU was running at a realtime priority over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYSCALL_TIME

The time, in seconds, that this CPU was running in system mode (not including interrupt, context switch, trap or vfault CPU) during the last interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYSCALL_TIME_CUM

The time, in seconds, that this CPU was running in system mode (not including interrupt, context switch, trap or vfault CPU) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYSCALL_UTIL

The percentage of time that this CPU was running in system mode (not including interrupt, context switch, trap or vfault CPU) during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYSCALL_UTIL_CUM

The average percentage of time that this CPU was running in system mode (not including interrupt, context switch, trap or vfault CPU) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_TIME

The time, in seconds, that this CPU (or logical processor) was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup.

Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_TIME_CUM

The time, in seconds, that this CPU (or logical processor) was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_UTIL

The percentage of time that this CPU (or logical processor) was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_UTIL_CUM

The percentage of time that this CPU (or logical processor) was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_TIME

The total time, in seconds, that this CPU (or logical processor) was not idle during the interval.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_TIME_CUM

The total time, in seconds, that this CPU (or logical processor) was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_UTIL

The percentage of time that this CPU (or logical processor) was not idle during the interval.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_UTIL_CUM

The average percentage of time that this CPU (or logical processor) was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TRAP_TIME

The time, in seconds, this CPU was in trap handler code during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TRAP_TIME_CUM

The time, in seconds, this CPU was in trap handler code over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TRAP_UTIL

The percentage of time this CPU was in trap handler code during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TRAP_UTIL_CUM

The average percentage of time this CPU was in trap handler code over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_USER_MODE_TIME

The time, in seconds, during the interval that this CPU (or logical processor) was in user mode.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_USER_MODE_TIME_CUM

The time, in seconds, that this CPU (or logical processor) was in user mode over the cumulative collection time. User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_USER_MODE_UTIL

The percentage of time that this CPU (or logical processor) was in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_USER_MODE_UTIL_CUM

The average percentage of time that this CPU (or logical processor) was in user mode over the cumulative collection time. User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_VFAULT_TIME

The time, in seconds, this CPU was handling page faults during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_VFAULT_TIME_CUM

The time, in seconds, this CPU was handling page faults over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_VFAULT_UTIL

The percentage of time this CPU was handling page faults during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_VFAULT_UTIL_CUM

The average percentage of time this CPU was handling page faults over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CSWITCH

The number of context switches for this CPU during the interval.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

BYCPU_CSWITCH_CUM

The number of context switches for this CPU over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

BYCPU_CSWITCH_RATE

The average number of context switches per second for this CPU during the interval.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

BYCPU_CSWITCH_RATE_CUM

The average number of context switches per second for this CPU over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

BYCPU_ID

The ID number of this CPU. On some Unix systems, such as SUN, CPUs are not sequentially numbered.

BYCPU_INTERRUPT

The number of device interrupts for this CPU during the interval.

On HP-UX, a value of “na” is displayed on a system with multiple CPUs.

BYCPU_INTERRUPT_RATE

The average number of device interrupts per second for this CPU during the interval.

On HP-UX, a value of “na” is displayed on a system with multiple CPUs.

BYCPU_INTERRUPT_STATE

A text string indicating whether the current processor is “enabled” or “disabled” for servicing IO interrupts.

BYCPU_LAST_PROC_ID

The process id (pid) of the last process to have used this CPU.

BYCPU_LAST_THREAD_ID

The thread ID (TID) number of the last kernel thread to have used this CPU.

BYCPU_LAST_USER_THREAD_ID

The user thread ID number of the last user thread to have used this CPU within the context of its associated process. A process may have multiple user threads. This indicates the most recently executed user thread of the process identified in BYCPU_LAST_PROC_ID.

BYCPU_RUN_QUEUE_15_MIN

This represents the 15 minute load average for this processor.

BYCPU_RUN_QUEUE_1_MIN

This represents the 1 minute load average for this processor.

BYCPU_RUN_QUEUE_5_MIN

This represents the 5 minute load average for this processor.

BYCPU_STATE

A text string indicating the current state of a processor.

On HP-UX, this is either “Enabled”, “Disabled” or “Unknown”. On AIX, this is either “Idle/Offline” or “Online”. On all other systems, this is either “Offline”, “Online” or “Unknown”.

BYDSKDETAIL_LABEL

The type of entry this disk device is associated with - could be a partition, file system directory, logical volume, or volume group.

BYDSKDETAIL_NAME

The name of the partition, file system directory, logical volume, or volume group this disk device is associated with.

BYDSK_AVG_QUEUE_TIME

The average time, in milliseconds, that a disk request spent waiting in the queue during the interval. For example, a value of 1.14 would indicate that disk requests during the last interval spent on average slightly longer than one-thousandths of a second wating in the queue of this device.

BYDSK_AVG_READ_QUEUE_TIME

The average time, in milliseconds, that a disk read request spent waiting in the queue during the interval. For example, a value of 1.14 would indicate that disk read requests during the last interval spent on average slightly longer than one-thousandths of a second waiting in the queue of this device.

BYDSK_AVG_READ_SERVICE_TIME

The average time, in milliseconds, that this disk device spent processing each disk read request during the interval. For example, a value of 5.14 would indicate that disk read requests during the last interval took on average slightly longer than five one-thousandths of a second to complete for this device.

This is a measure of the speed of the disk, because slower disk devices typically show a larger average read service time. Average read service time is also dependent on factors such as the distribution of I/O requests over the interval and their locality. It can also be influenced by disk driver and controller features such as I/O merging and command queueing. Note that this write service time is measured from the perspective of the kernel, not the disk device itself. For example, if a disk device can find the requested data in its cache, the average service time could be quicker than the speed of the physical disk hardware.

This metric can be used to help determine which disk devices are taking more time than usual to process read requests.

BYDSK_AVG_SERVICE_TIME

The average time, in milliseconds, that this disk device spent processing each disk request during the interval. For example, a value of 5.14 would indicate that disk requests during the last interval took on average slightly longer than five one-thousandths of a second to complete for this device.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

This is a measure of the speed of the disk, because slower disk devices typically show a larger average service time. Average service time is also dependent on factors such as the distribution of I/O requests over the interval and their locality. It can also be influenced by disk driver and controller features such as I/O merging and command queueing. Note that this service time is measured from the perspective of the kernel, not the disk device itself. For example, if a disk device can find the requested data in its cache, the average service time could be quicker than the speed of the physical disk hardware.

This metric can be used to help determine which disk devices are taking more time than usual to process requests.

BYDSK_AVG_WRITE_QUEUE_TIME

The average time, in milliseconds, that a disk write request spent waiting in the queue during the interval. For example, a value of 1.14 would indicate that disk write requests during the last interval spent on average slightly longer than one-thousandths of a second waiting in the queue of this device.

BYDSK_AVG_WRITE_SERVICE_TIME

The average time, in milliseconds, that this disk device spent processing each disk write request during the interval. For example, a value of 5.14 would indicate that disk write requests during the last interval took on average slightly longer than five one-thousandths of a second to complete for this device.

This is a measure of the speed of the disk, because slower disk devices typically show a larger average write service time. Average write service time is also dependent on factors such as the distribution of I/O requests over the interval and their locality. It can also be influenced by disk driver and controller features such as I/O merging and command queueing. Note that this write service time is measured from the perspective of the kernel, not the disk device itself. For example, if a disk device can find the requested data in its cache, the average service time could be quicker than the speed of the physical disk hardware.

This metric can be used to help determine which disk devices are taking more time than usual to process write requests.

BYDSK_BUS

The name of the bus interface used by this disk.

BYDSK_BUSY_TIME

The time, in seconds, that this disk device was busy transferring data during the interval.

On HP-UX, this is the time, in seconds, during the interval that the disk device had IO in progress from the point of view of the Operating System. In other words, the time, in seconds, the disk was busy servicing requests for this device.

BYDSK_CONTROLLER

The disk controller name. This information is only available for disks using the hpib or hpfl interfaces.

BYDSK_DEVNAME

The name of this disk device.

On HP-UX, the name identifying the specific disk spindle is the hardware path which specifies the address of the hardware components leading to the disk device.

On SUN, these names are the same disk names displayed by “iostat”.

On AIX, this is the path name string of this disk device. This is the fsname parameter in the mount(1M) command. If more than one file system is contained on a device (that is, the device is partitioned), this is indicated by an asterisk (“*”) at the end of the path name.

On OSF1, this is the path name string of this disk device. This is the file-system parameter in the mount(1M) command.

On Windows, this is the unit number of this disk device.

BYDSK_DEVNO

Major / Minor number of the device.

BYDSK_DIRNAME

The name of the file system directory mounted on this disk device. If more than one file system is mounted on this device, “Multiple FS” is seen.

BYDSK_DISKNAME

The device special file(DSF) representing this disk. This metric only gives the last component in the DSF path.

On HP-UX 11iv1 and 11iv2, the DSF is of the form /dev/dsk/c#t#d# and hence value of DISKNAME metric will be “c#t#d#”

On HP-UX 11iv3, this metric gives the path independent DSF name. So value of DISKNAME metric will be “disk#”. See intro(7) for more details.

BYDSK_FS_IO_RATE

The number of physical file system reads and writes per second to this disk device during the interval.

BYDSK_FS_READ

The number of physical file system reads from this disk device during the interval.

BYDSK_FS_READ_RATE

The number of physical file system reads per second from this disk device during the interval.

BYDSK_FS_WRITE

The number of physical file system writes to this disk device during the interval.

BYDSK_FS_WRITE_RATE

The number of physical file system writes per second to this disk device during the interval.

BYDSK_ID

The ID of the current disk device.

BYDSK_INTERVAL

The amount of time in the interval.

BYDSK_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYDSK_LOGL_BYTE_RATE

The number of logical read or write KBs per second to this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_BYTE_RATE_CUM

The average number of KBs of logical read or writes to this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_IO_RATE

The total number of logical IOs per second for this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_IO_RATE_CUM

The average number of logical IOs per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_READ

The number of logical reads for this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_READ_BYTE_RATE

The number of logical read KBs per second from this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_READ_BYTE_RATE_CUM

The average number of logical KBs per second read from this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_READ_RATE

The number of logical reads per second for this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_READ_RATE_CUM

The average number of logical reads per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_WRITE

The number of logical writes for this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_WRITE_BYTE_RATE

The number of logical writes KBs per second to this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_WRITE_BYTE_RATE_CUM

The average number of KBs of logical writes per second to this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_WRITE_RATE

The number of logical writes per second for this disk device during the interval.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_LOGL_WRITE_RATE_CUM

The average number of logical writes per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the logical IO rates by disk device cannot be obtained in a multi-disk LVM configuration because there is no reasonable means of tying logical IO transactions to physical spindles spanned on the logical volume. Therefore, if you have a multi-disk LVM configuration, you always see “na” for this metric.

BYDSK_PHYS_BYTE

The number of KBs of physical IOs transferred to or from this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_BYTE_RATE

The average KBs per second transferred to or from this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_BYTE_RATE_CUM

The average number of KBs per second of physical reads and writes to or from this disk device over the cumulative collection time.

On Unix systems, this includes all types of physical disk IOs including file system, virtual memory, and raw IOs.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYDSK_PHYS_IO

The number of physical IOs for this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw reads.

BYDSK_PHYS_IO_RATE

The average number of physical IO requests per second for this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system IO, virtual memory and raw IO.

BYDSK_PHYS_IO_RATE_CUM

The average number of physical reads and writes per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYDSK_PHYS_READ

The number of physical reads for this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On AIX, this is an estimated value based on the ratio of read bytes to total bytes transferred. The actual number of reads is not tracked by the kernel. This is calculated as

```
BYDSK_PHYS_READ =
  BYDSK_PHYS_IO *
  (BYDSK_PHYS_READ_BYTE /
   BYDSK_PHYS_IO_BYTE)
```

BYDSK_PHYS_READ_BYTE

The KBs transferred from this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_READ_BYTE_RATE

The average KBs per second transferred from this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_READ_BYTE_RATE_CUM

The average number of KBs per second of physical reads from this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYDSK_PHYS_READ_RATE

The average number of physical reads per second for this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On AIX, this is an estimated value based on the ratio of read bytes to total bytes transferred. The actual number of reads is not tracked by the kernel. This is calculated as

```
BYDSK_PHYS_READ_RATE =
  BYDSK_PHYS_IO_RATE *
```

```
      (BYDSK_PHYS_READ_BYTE /
       BYDSK_PHYS_IO_BYTE)
```

BYDSK_PHYS_READ_RATE_CUM

The average number of physical reads per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYDSK_PHYS_WRITE

The number of physical writes for this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On AIX, this is an estimated value based on the ratio of write bytes to total bytes transferred because the actual number of writes is not tracked by the kernel. This is calculated as

```
  BYDSK_PHYS_WRITE =
    BYDSK_PHYS_IO *
    (BYDSK_PHYS_WRITE_BYTE /
     BYDSK_PHYS_IO_BYTE)
```

BYDSK_PHYS_WRITE_BYTE

The KBs transferred to this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_WRITE_BYTE_RATE

The average KBs per second transferred to this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_WRITE_BYTE_RATE_CUM

The average number of KBs per second of physical writes to this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYDSK_PHYS_WRITE_RATE

The average number of physical writes per second for this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On AIX, this is an estimated value based on the ratio of write bytes to total bytes transferred. The actual number of writes is not tracked by the kernel. This is calculated as

```
BYDSK_PHYS_WRITE_RATE =
  BYDSK_PHYS_IO_RATE *
  (BYDSK_PHYS_WRITE_BYTE /
   BYDSK_PHYS_IO_BYTE)
```

BYDSK_PHYS_WRITE_RATE_CUM

The average number of physical writes per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYDSK_PRODUCT_ID

The disk product ID.

BYDSK_QUEUE_0_UTIL

The percentage of intervals during which there were no IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 1.5, 0, and 3, then the value for this metric would be 50% since 50% of the intervals had a zero queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_2_UTIL

The percentage of intervals during which there were 1 or 2 IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 1, 0, and 2, then the value for this metric would be 50% since 50% of the intervals had a 1-2 queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_4_UTIL

The percentage of intervals during which there were 3 or 4 IO requests waiting to use this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 3, 0, and 4, then the value for this metric would be 50% since 50% of the intervals had a 3-4 queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_8_UTIL

The percentage of intervals during which there were between 5 and 8 IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 8, 0, and 5, then the value for this metric would be 50% since 50% of the intervals had a 5-8 queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_X_UTIL

The percentage of intervals during which there were more than 8 IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 9, 0, and 10, then the value for this metric would be 50% since 50% of the intervals had queue length greater than 8.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_RAW_IO_RATE

The number of raw reads or writes per second made to this disk device during the interval.

BYDSK_RAW_READ

The number of physical raw reads made from this disk device during the interval.

BYDSK_RAW_READ_RATE

The number of raw reads per second made from this disk device during the interval.

BYDSK_RAW_WRITE

The number of physical raw writes made to this disk device during the interval.

BYDSK_RAW_WRITE_RATE

The number of raw writes per second made to this disk device during the interval.

BYDSK_REQUEST_QUEUE

The average number of IO requests that were in the wait queue for this disk device during the interval. These requests are the physical requests (as opposed to logical IO requests).

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_SYSTEM_IO

The number of physical system reads or writes to this disk device during the interval.

BYDSK_SYSTEM_IO_RATE

The number of physical system reads or writes per second to this disk device during the interval.

BYDSK_SYSTEM_READ_RATE

The number of physical system reads per second from this disk device during the interval.

BYDSK_SYSTEM_WRITE_RATE

The number of physical system writes per second to this disk device during the interval.

BYDSK_TIME

The time of day of the interval.

BYDSK_UTIL

On HP-UX, this is the percentage of the time during the interval that the disk device had IO in progress from the point of view of the Operating System. In other words, the utilization or percentage of time busy servicing requests for this device.

On the non-HP-UX systems, this is the percentage of the time that this disk device was busy transferring data during the interval.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

This is a measure of the ability of the IO path to meet the transfer demands being placed on it. Slower disk devices may show a higher utilization with lower IO rates than faster disk devices such as disk arrays. A value of greater than 50% utilization over time may indicate that this device or its IO path is a bottleneck, and the access pattern of the workload, database, or files may need reorganizing for better balance of disk IO load.

BYDSK_UTIL_CUM

On HP-UX, this is the percentage of the time that this disk device had IO in progress from the point of view of the Operating System over the cumulative collection time. In other words, this is the utilization or percentage of time busy servicing requests for this device.

On all other Unix systems, this is the percentage of the time that this disk device was busy transferring data over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

This is a measure of the ability of the IO path to meet the transfer demands being placed on it. Slower disk devices may show a higher utilization with lower IO rates than faster disk devices such as disk arrays. A value of greater than 50% utilization over time may indicate that this device or its IO path is a bottleneck, and the access pattern of the workload, database, or files may need reorganizing for better balance of disk IO load.

BYDSK_VENDOR_ID

The disk vendor ID. This information is only available for disks using the scsi interface.

BYDSK_VM_IO

The number of virtual memory IOs to this disk device during the interval.

BYDSK_VM_IO_RATE

The number of virtual memory IOs per second to this disk device during the interval.

BYDSK_VM_READ_RATE

The number of virtual memory reads per second from this disk device during the interval.

BYDSK_VM_WRITE_RATE

The number of virtual memory writes per second to this disk device during the interval.

BYHBA_AVG_SERVICE_TIME

This is the average time, in milli seconds, this device took to service one request. This metric is supported on HP-UX 11iv3 and above.

BYHBA_AVG_WAIT_TIME

This is the time, in milli seconds, that a request had to wait in the device queue before getting processed. This metric is supported on HP-UX 11iv3 and above.

BYHBA_BUSY_TIME

This is the time in seconds, during the interval that the device had IO in progress from the point of view of the Operating System. In other words, the time, in seconds, the device was busy servicing requests. This metric is supported on HP-UX 11iv3 and above.

BYHBA_BYTE_RATE

The average KBs per second transferred to or from this card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_BYTE_RATE_CUM

The average KBs per second transferred to or from this card during the cumulative collection interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_CLASS

The class of the Host Bus Adaptor. This metric is supported on HP-UX 11iv3 and above.

BYHBA_DEVNAME

The hardware path of the Host Bus Adaptor. This metric is supported on HP-UX 11iv3 and above.

BYHBA_DEVNO

Major / Minor number of the device. This metric is supported on HP-UX 11iv3 and above.

BYHBA_DRIVER

Name of driver handling the Host Bus Adaptor. This metric is supported on HP-UX 11iv3 and above.

BYHBA_ID

The instance number of the Host Bus Adaptor. This metric is supported on HP-UX 11iv3 and above.

BYHBA_INTERVAL

The amount of time in the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_INTERVAL_CUM

The amount of time over the cumulative collection time. This metric is supported on HP-UX 11iv3 and above.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYHBA_IO

Number of IO requests handled by the HBA in this interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_IO_RATE

The average number of IO requests per second for this device during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_IO_RATE_CUM

The average number of reads and writes per second for this card over the cumulative collection time. This metric is supported on HP-UX 11iv3 and above.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYHBA_NAME

The name of the Host Bus Adaptor. This metric is supported on HP-UX 11iv3 and above.

BYHBA_READ

The number of reads for this IO card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_READ_BYTE_RATE

The average KBs per second read from this card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_READ_BYTE_RATE_CUM

The average KBs per second read from this card during the cumulative collection interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_READ_RATE

The number of reads for this IO card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_READ_RATE_CUM

The average number of reads per second for this card over the cumulative collection time. This metric is supported on HP-UX 11iv3 and above.

BYHBA_REQUEST_QUEUE

The average number of IO requests that were in the wait queue for this disk device during the interval. These requests are the physical requests (as opposed to logical IO requests). This metric is supported on HP-UX 11iv3 and above.

BYHBA_STATE

The state of the Host Bus Adaptor(“Active”/“Closed”). This metric is supported on HP-UX 11iv3 and above.

BYHBA_THROUGHPUT_UTIL

Percentage of IO bandwidth utilized by the Host Bus Adaptor. This metric is supported on HP-UX 11iv3 and above.

BYHBA_TIME

The time of day of the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_TYPE

The type of device. “HBA” for HBA card and “TAPE” for tape drives. This metric is supported on HP-UX 11iv3 and above.

BYHBA_UTIL

Percentage of time HBA was busy servicing the IO requests in this interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_WRITE

The number of writes for this IO card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_WRITE_BYTE_RATE

The average KBs per second written to this card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_WRITE_BYTE_RATE_CUM

The average KBs per second written to this card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_WRITE_RATE

The number of writes for this IO card during the interval. This metric is supported on HP-UX 11iv3 and above.

BYHBA_WRITE_RATE_CUM

The average number of writes per second for this card over the cumulative collection time. This metric is supported on HP-UX 11iv3 and above.

BYLS_CPU_CYCLE_ENTL_MAX

On vMA, for a host, logical system and resource pool this value indicates the maximum processor capacity, in MHz, configured for the entity. If the maximum processor capacity is not configured for the entity, a value of “-3” will be displayed in PA and “ul”(unlimited) in other clients.

On HPUX, the maximum processor capacity, in MHz, configured for this logical system.

BYLS_CPU_CYCLE_ENTL_MIN

On vMA, for a host, logical system and resource pool this value indicates the minimum processor capacity, in MHz, configured for the entity.

On HPUX, the minimum processor capacity, in MHz, configured for this logical system.

BYLS_CPU_ENTL_MAX

The maximum CPU units configured for a logical system.

On HP-UX HPVM, this metric indicates the maximum percentage of physical CPU that a virtual CPU of this logical system can get.

On AIX SPLPAR, this metric is equivalent to “Maximum Capacity“ field of 'lparstat -i' command.

For WPARs, it is the maximum percentage of CPU that a WPAR can have even if there is no contention for CPU. WPAR shares CPU units of its global environment.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a host, the metric is equivalent to total number of cores on the host. For a resource pool and a logical system, this metrics indicates the maximum CPU units configured for it.

BYLS_CPU_ENTL_MIN

The minimum CPU units configured for this logical system.

On HP-UX HPVM, this metric indicates the minimum percentage of physical CPU that a virtual CPU of this logical system is guaranteed.

On AIX SPLPAR, this metric is equivalent to “Minimum Capacity“ field of 'lparstat -i' command.

For WPARs, it is the minimum CPU share assigned to a WPAR that is guaranteed. WPAR shares CPU units of its global environment.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a host, the metric is equivalent to total number of cores on the host. For a resource pool and a logical system, this metrics indicates the guranteed minimum CPU units configured for it.

On Solaris Zones, this metrics indicates the configured minimum CPU percentage reserved for a logical system.

For Solaris Zones, this metric is calculated as:

BYLS_CPU_ENTL_MIN = (BYLS_CPU_SHARES_PRIO / Pool-Cpu-Shares)

where, Pool-Cpu-Shares is the total CPU shares available with CPU pool the zone is associated with. Pool-Cpu-Shares is addition of BYLS_CPU_SHARES_PRIO values for all active zones associated with this pool.

BYLS_CPU_ENTL_UTIL

Percentage of entitled processing units (guaranteed processing units allocated to this logical system) consumed by the logical system.

On a HP-UX HPVM host the metric indicates the logical system's CPU utilization with respect to minimum CPU entitlement.

On HP-UX HPVM host, this metric is calculated as: BYLS_CPU_ENTL_UTIL = (BYLS_CPU_PHYSC / (BYLS_CPU_ENTL_MIN * BYLS_NUM_CPU)) * 100

On AIX, this metric is calculated as: BYLS_CPU_ENTL_UTIL = (BYLS_CPU_PHYSC / BYLS_CPU_ENTL) * 100

On WPAR, this metric is calculated as: BYLS_CPU_ENTL_UTIL = (BYLS_CPU_PHYSC / BYLS_CPU_ENTL_MAX) * 100 This metric matches “%Resc” of topas command (inside WPAR)

On Solaris Zones, the metric indicates the logical system's CPU utilization with respect to minimum CPU entitlement. This metric is calculated as:

BYLS_CPU_ENTL_UTIL = (BYLS_CPU_TOTAL_UTIL / BYLS_CPU_SHARES_PRIO) * 100

If a Solaris zone is not assigned a CPU entitlement value then a CPU entitlement value is derived for this zone based on total CPU entitlement associated with the CPU pool this zone is attached to.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a host the value is same as BYLS_CPU_PHYS_TOTAL_UTIL while for logical system and resource pool the value is the percentage of processing units consumed w.r.t minimum CPU entitlement.

BYLS_CPU_PHYSC

This metric indicates the number of CPU units utilized by the logical system.

On an Uncapped logical system, this value will be equal to the CPU units capacity used by the logical system during the interval. This can be more than the value entitled for a logical system.

BYLS_CPU_PHYS_TOTAL_TIME

Total time in seconds, spent by the logical system on the physical CPUs.

On HPUX, this information is updated internally every 10 seconds so it may take that long for these values to be updated in PA/Glance.

On vMA, the value indicates the time spent in seconds on the physical CPU. by logical system or host or resource pool,

BYLS_CPU_PHYS_TOTAL_TIME_CUM

Total time in seconds, spent by the logical system on physical CPUs, from the start of measurement.

BYLS_CPU_PHYS_TOTAL_UTIL

Percentage of total time the physical CPUs were utilized by this logical system during the interval.

On HPUX, this information is updated internally every 10 seconds so it may take that long for these values to be updated in PA/Glance.

On Solaris, this metric is calculated with respect to the available active physical CPUs on the system.

On AIX, this metric is equivalent to sum of BYLS_CPU_PHYS_USER_MODE_UTIL and BYLS_CPU_PHYS_SYS_MODE_UTIL.

For AIX lpars, the metric is calculated with respect to the available physical CPUs in the pool to which this LPAR belongs to.

For AIX wpars, the metric is calculated with respect to the available physical CPUs in the resource set or Global Environment.

On vMA, the value indicates percentage of total time the physical CPUs were utilized by logical system or host or resource pool,

BYLS_CPU_TOTAL_UTIL

Percentage of total time the logical CPUs were not idle during this interval.

This metric is calculated against the number of logical CPUs configured for this logical system.

For AIX wpars, the metric represents the percentage of time the physical CPUs were not idle during this interval.

BYLS_DISPLAY_NAME

On vMA, this metric indicates the name of the host or logical system or resource pool.

On HPVM, this metric indicates the Virtual Machine name of the logical systemand is equivalent to “Virtual Machine Name” field of 'hpvmstatus' command.

On AIX the value is as returned by the command “uname -n” (that is, the string returned from the “hostname” program).

On Solaris Zones, this metric indicates the zone name and is equivalent to 'NAME' field of 'zoneadm list -vc' command.

On Hyper-V host, this metric indicates the Virtual Machine name of the logical systemand is equivalent to the Name displayed in Hyper-V Manager. For Root partition, the value is always “Root”.

BYLS_IP_ADDRESS

This metric indicates IP Address of the particular logical system.

On vMA, this metric indicates the IP Address for a host and a logical system while for a resource pool the value is NA.

BYLS_LS_HOSTNAME

This is the DNS registered name of the system.

On Hyper-V host, this metric is NA if the logical system is not active or Hyper-V Integration Components are not installed on it.

On vMA, for a host and logical system the metric is the Fully Qualified Domain Name, while for resource pool the value is NA.

BYLS_LS_ID

An unique identifier of the logical system.

On HPVM, this metric is a numeric id and is equivalent to “VM # ” field of 'hpvmstatus' command.

On AIX LPAR, this metric indicates partition number and is equivalent to “Partition Number” field of 'lparstat -i' command. For aix wpar, this metric represents the partition number and is equivalent to “uname -W” from inside wpar.

On Solaris Zones, this metric indicates the zone id and is equivalent to 'ID' field of 'zoneadm list -vc' command.

On Hyper-V host, this metric indicates the PID of the process corresponding to this logical system. For Root partition, this metric is NA.

On vMA, this metric is a unique identifier for a host, resource pool and a logical system. The value of this metric may change for an instance across collection intervals.

BYLS_LS_MODE

This metric indicates whether the CPU entitlement for the logical system is Capped or Uncapped.

On AIX SPLPAR, this metric is same as “Mode” field of 'lparstat -i' command.

For WPARs, this metric is always CAPPED.

On vMA, the value is Capped for a host and Uncapped for a logical system. For resource pool, the value is Uncapped or Capped depending on whether the reservation is expandable or not for it.

On Solaris Zones, this metric is “Capped” when the zone is assigned CPU shares and is attached to a valid CPU pool.

BYLS_LS_NAME

This is the name of the computer.

On HPVM, this metric indicates the Virtual Machine name of the logical systemand is equivalent to “Virtual Machine Name” field of 'hpvmstatus' command.

On AIX the value is as returned by the command “uname -n” (that is, the string returned from the “hostname” program).

On vMA, this metric is a unique identifier for host, resource pool and a logical system. The value of this metric remains the same, for an instance, across collection intervals.

On Solaris Zones, this metric indicates the zone name and is equivalent to 'NAME' field of 'zoneadm list -vc' command.

On Hyper-V host, this metric indicates the name of the XML file which has configuration information of the logical system. This file will be present under the logical system's installation directory indicated by BYLS_LS_PATH. For Root partition, the value is always “Root”.

BYLS_LS_OSTYPE

The Guest OS this logical system is hosting.

On HPVM, the metric can have following values: HP-UX Linux Windows OpenVMS Other Unknown

On Hyper-V host, the metric can have following values: Windows Other

On Hyper-V host, this metric is NA if the logical system is not active or Hyper-V Integration Components are not installed on it.

On vMA, the metric can have the following values for host and logical system: ESX/ESXi followed by version or ESX-Serv (applicable only for a host) Linux Windows Solaris Unknown The value is NA for resource pool

BYLS_LS_PROC_ID

On HPVM host and Hyper-V host, each VM is manifested as a process. These processes have the executable name hpvmapp for HPVM and vmwp.exe for Hyper-V host. This metric will have the PID of the process corresponding to this logical system.

On HPVM, typically hpvmapp has the option -d whose argument is the name of the VM.

On Hyper-V host, for Root partition, this metric is NA.

BYLS_LS_SHARED

This metric indicates whether the physical CPUs are dedicated to this logical system or shared.

On HPUX HPVM, and Hyper-V host,this metric is always “Shared”.

On vMA, the value is “Dedicated” for host, and “Shared” for logical system and resource pool.

On AIX SPLPAR, this metric is equivalent to “Type” field of 'lparstat -i' command. For AIX wpars,this metric will be always “Shared”.

On Solaris Zones, this metric is “Dedicated” when this zone is attached to a CPU pool not shared by any other zone.

BYLS_LS_STATE

The state of this logical system.

On HPVM, the logical systems can have one of the following states: Unknown Other invalid Up Down Boot Crash Shutdown Hung

On vMA, this metric can have one of the following states for a host: on off unknown The values for a logical system can be one of the following: on off suspended unknown The value is NA for resource pool.

On Solaris Zones, the logical systems can have one of the following states: configured incomplete installed ready running shutting down mounted

On AIX lpars, the logical system will be always active. On AIX wpars, the logical systems can have one of the following states: Broken Transitional Defined Active Loaded Paused Frozen Error

A logical system on a Hyper-V host can have the following states: unknown enabled disabled paused suspended starting snapshtng migrating saving stopping deleted pausing resuming

BYLS_LS_UUID

UUID of this logical system. This Id uniquely identifies this logical system across multiple hosts.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a logical system or a host, the value indicates the UUID appended to display_name of the system. For a resource pool the value is hostname of the host where resource pool is hosted followed by the unique id of resource pool.

BYLS_MEM_ENTL

The entitled memory configured for this logical system (in MB).

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for host the value is the physical memory available in the system and for logical system this metric indicates the minimum memory configured while for resource pool the value is NA.

BYLS_MEM_ENTL_MAX

The maximum amount of memory configured for a logical system, in MB.

The value of this metric will be “-3” in PA and “ul” in other clients if entitlement is 'Unlimited' for a logical system.

On AIX LPARs, this metric will be “na”.

On vMA, this metric indicates the maximum amount of memory configured for a resource pool or a logical system. For a host, the value is the amount of physical memory available in the system.

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”.

BYLS_MEM_ENTL_MIN

The minimum amount of memory configured for the logical system, in MB.

On AIX LPARs, this metric will be “na”.

On vMA, this metric indicates the reserved amount of memory configured for a host, resource pool or a logical system.

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”.

BYLS_MEM_ENTL_UTIL

The percentage of entitled memory in use during the interval.

On vMA, for a logical system or a host, the value indicates percentage of entitled memory in use during the interval by it. On vMA, for a resource pool, this metric is “na”.

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”.

BYLS_MEM_FREE

The amount of free memory on the logical system, in MB.

On vMA, for a host and logical system, it is the amount of memory not allocated. For a resource pool the value is “na”.

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”.

BYLS_MEM_FREE_UTIL

The percentage of memory that is free at the end of the interval.

On vMA, for a resource pool the value is NA.

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”.

BYLS_MEM_HEALTH

On vMA, for a host, it is a number that indicates the state of the memory. Low number indicates system is not under memory pressure. For a logical system and resource pool the value is “na”.

On vMA, the values are defined as:

```
  0 - High - indicates free memory is available and no memory
pressure.
  1 - Soft
  2 - Hard
  3 - Low  - indicates there is a pressure for free memory.
```

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”. For relevant guests, these values represent the level of memory pressure, 0 being none and 3 being very high.

BYLS_MEM_PHYS

On vMA, for host the value is the physical memory available in the system and for logical system this metric indicates the minimum memory configured. On vMA, for a resource pool, this metric is “na”.

On HPVM, this metric matches the data in the “Memory Details” section of “hpvmstatus -V”, when the dynamic memory driver is not enabled, and it matches the data in the “Dynamic Memory Information” section when the dynamic memory driver is active. The dynamic memory driver is currently only available on guests running HPUX 11iv3 or newer versions.

BYLS_MEM_PHYS_UTIL

The percentage of physical memory used during the interval.

On vMA, the metric indicates the percentage of physical memory used by a host, logical system.

On vMA, for a resource pool, this metric is “na”.

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”.

BYLS_MEM_USED

The amount of memory used by the logical system at the end of the interval.

On vMA, this applies to hosts, resource pools and logical systems.

On vMA, for a resource pool, this metric is “na”.

On HPVM, this metric is valid for HPUX guests running 11iv3 or newer releases, with the dynamic memory driver active. Running “hpvmstatus -V” will indicate whether the driver is active. For all other guests, the value is “na”.

BYLS_NUM_CPU

The number of virtual CPUs configured for this logical system. This metric is equivalent to GBL_NUM_CPU on the corresponding logical system.

On HPVM, the maximum CPUs a logical system can have is 4 with respect to HPVM 3.x.

On AIX SPLPAR, the number of CPUs can be configured irrespective of the available physical CPUs in the pool this logical system belongs to. For AIX wpars, this metric represents the logical CPUs of the global environment.

On vMA, for a host the metric is the number of physical CPU threads on the host. For a logical system, the metric is the number of virtual cpus configured.For a resource pool the metric is NA.

On Solaris Zones, this metric represents number of CPUs in the CPU pool this zone is attached to. This metric value is equivalent to GBL_NUM_CPU inside corresponding non-global zone.

BYLS_NUM_DISK

The number of disks configured for this logical system. Only local disk devices and optical devices present on the system are counted in this metric.

On vMA, for a host the metric is the number of disks configured for the host . For a logical system, the metric is the number of logical disk devices present on the logical system. For a resource pool the metric is NA.

For AIX wpars, this metric will be “na”.

On Hyper-V host, this metric value is equivalent to GBL_NUM_DISK inside corresponding Hyper-V guest.

On Hyper-V host, this metric is NA if the logical system is not active.

BYLS_NUM_NETIF

The number of network interfaces configured for this logical system.

On LPAR, this metric includes the loopback interface.

On Hyper-V host, this metric value is equivalent to GBL_NUM_NETWORK inside corresponding Hyper-V guest.

On Solaris Zones, this metric value is equivalent to GBL_NUM_NETWORK inside corresponding non-global zone.

On Hyper-V host, this metric is NA if the logical system is not active.

On vMA, for a host the metric is the number of network adapters on the host. For a logical system, the metric is the number of network interfaces configured for the logical system. For a resource pool the metric is NA.

BYLS_UPTIME_SECONDS

The uptime of this logical system in seconds.

On AIX LPARs, this metric will be “na”.

On vMA, for a host and logical system the metric is the uptime in seconds while for a resource pool the metric is NA.

BYNETIF_COLLISION

The number of physical collisions that occurred on the network interface during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

For HP-UX, this will be the same as the sum of the “Single Collision Frames“, ”Multiple Collision Frames“, ”Late Collisions“, and ”Excessive Collisions“ values from the output of the ”lanadmin“ utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For most other Unix systems, this is the same as the sum of the “Coll” column from the “netstat -i” command (“collisions” from the “netstat -i -e” command on Linux) for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric will be N/A.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_COLLISION_1_MIN_RATE

The number of physical collisions per minute on the network interface during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_COLLISION_RATE

The number of physical collisions per second on the network interface during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_COLLISION_RATE_CUM

The average number of physical collisions per second on the network interface over the cumulative collection time. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_ERROR

The number of physical errors that occurred on the network interface during the interval. An increasing number of errors may indicate a hardware problem in the network.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

For HP-UX, this will be the same as the sum of the “Inbound Errors” and “Outbound Errors” values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of “Ierrs” (RX-ERR on Linux) and “Oerrs” (TX-ERR on Linux) from the “netstat -i” command for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_ERROR_1_MIN_RATE

The number of physical errors per minute on the network interface during the interval.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_ERROR_RATE

The number of physical errors per second on the network interface during the interval.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_ERROR_RATE_CUM

The average number of physical errors per second on the network interface over the cumulative collection time.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric will be N/A.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_ID

The ID number of the network interface.

BYNETIF_INTERVAL

The amount of time in the interval.

BYNETIF_INTERVAL_CUM

The amount of time over the cumulative collection time.

BYNETIF_IN_BYTE

The number of KBs received from the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_BYTE_RATE

The number of KBs per second received from the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_BYTE_RATE_CUM

The average number of KBs per second received from the network via this interface over the cumulative collection time. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_PACKET

The number of successful physical packets received through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

For HP-UX, this will be the same as the sum of the “Inbound Unicast Packets“ and ”Inbound Non-Unicast Packets“ values from the output of the “lanadmin” utility for the network interface.

Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the “Ipkts” column (RX-OK on Linux) from the “netstat -i” command for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_PACKET_RATE

The number of successful physical packets per second received through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_PACKET_RATE_CUM

The average number of physical packets per second received through the network interface over the cumulative collection time.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_LOGL_INTERVAL

The amount of time in the interval.

On systems with large numbers of Interface Protocol (IP) addresses, the measurement code now dynamically determines the interval for updating the BYNETIF_LOGL_* metrics. This reduces the collection overhead for these metrics in Glance and GPM. For the interval, it looks at how many IP addresses there are. The update interval for the BYNETIF_LOGL_* metrics is then set as follows:

```
* For 1 - 20 IP addresses, the
  counters are updated at the normal
  sampling interval.

* For 21 - 120 IP addresses, the
  counters are updated at an
  interval (in seconds) equal to the
  number of IP addresses.

* For more than 120 IP addresses,
  the counters are updated every 120
  seconds.
```

For example, if Glance or GPM is run with 5-second update intervals on an 11.0 system with 200 IP addresses configured, the information shown in the Network detail screens will only change once every 2 minutes. The data reflects all activity over that time so no information is lost.

BYNETIF_LOGL_INTERVAL_CUM

The amount of time over the cumulative collection time.

On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_LOGL_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_LOGL_INTERVAL can be greater than the sampling interval.

BYNETIF_LOGL_IN_PACKET

The number of successful logical packets received through the logical interface during the interval.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This is the same as the “Ipkts” column from the “netstat -i” command for a network device. See also netstat(1).

On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_LOGL_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_LOGL_INTERVAL can be greater than the sampling interval.

BYNETIF_LOGL_IN_PACKET_RATE

The number of successful logical packets per second received through the logical interface during the interval.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_LOGL_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_LOGL_INTERVAL can be greater than the sampling interval.

BYNETIF_LOGL_IN_PACKET_RATE_CUM

The average number of logical packets per second received through the logical interface over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_LOGL_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_LOGL_INTERVAL can be greater than the sampling interval.

BYNETIF_LOGL_IP_ADDRESS

The Internet IP address of this logical network interface. See also netstat(1).

BYNETIF_LOGL_NAME

The name of the logical network interface. These are the same names that appear in the “Name” column of the “netstat -i” command output.

BYNETIF_LOGL_OUT_PACKET

The number of successful logical packets sent through the logical interface during the interval.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This is the same as the “Opkts” column from the “netstat -i” command for a network device. See also netstat(1).

On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_LOGL_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_LOGL_INTERVAL can be greater than the sampling interval.

BYNETIF_LOGL_OUT_PACKET_RATE

The number of successful logical packets per second sent through the logical interface during the interval.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_LOGL_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_LOGL_INTERVAL can be greater than the sampling interval.

BYNETIF_LOGL_OUT_PACKET_RATE_CUM

The average number of logical packets per second sent through the logical interface over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_LOGL_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_LOGL_INTERVAL can be greater than the sampling interval.

BYNETIF_NAME

The name of the network interface.

For HP-UX 11.0 and beyond, these are the same names that appear in the “Description” field of the “lanadmin” command output.

On all other Unix systems, these are the same names that appear in the “Name” column of the “netstat -i” command.

Some examples of device names are:

```
  lo   - loop-back driver
  ln   - Standard Ethernet driver
  en   - Standard Ethernet driver
  le   - Lance Ethernet driver
  ie   - Intel Ethernet driver
  tr   - Token-Ring driver
  et   - Ether Twist driver
  bf   - fiber optic driver
```

All of the device names will have the unit number appended to the name. For example, a loop-back device in unit 0 will be “lo0”.

On vMA for Lan cards which are of type ESXVLan, this metric contains the vmnic<number> as first half and the second half is the ESX host name.

BYNETIF_NET_MTU

The size of the maximum transfer unit (MTU) for this interface.

BYNETIF_NET_SPEED

The speed of this interface. This is the bandwidth in Mega bits/sec.

BYNETIF_NET_TYPE

The type of network device the interface communicates through.

```
Lan     - local area network card
Loop    - software loopback
          interface (not tied to a
          hardware device)
Loop6   - software loopback
          interface IPv6 (not tied
          to a hardware device)
Serial  - serial modem port
Vlan    - virtual lan
Wan     - wide area network card
Tunnel  - tunnel interface
Apa     - HP LinkAggregate Interface (APA)
Other   - hardware network interface
          type is unknown.
ESXVLan - The card type belongs to network cards of ESX hosts which
are
            monitored on vMA.
```

BYNETIF_OUT_BYTE

The number of KBs sent to the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_BYTE_RATE

The number of KBs per second sent to the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_BYTE_RATE_CUM

The average number of KBs per second sent to the network via this interface over the cumulative collection time. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_PACKET

The number of successful physical packets sent through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

For HP-UX, this will be the same as the sum of the “Outbound Unicast Packets“ and ”Outbound Non-Unicast Packets“ values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the “Opkts” column (TX-OK on Linux) from the “netstat -i” command for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_PACKET_RATE

The number of successful physical packets per second sent through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_PACKET_RATE_CUM

The average number of successful physical packets per second sent through the network interface over the cumulative collection time.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_PACKET_RATE

The number of successful physical packets per second sent and received through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

If BYNETIF_NET_TYPE is “ESXVLan”, then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show “na” for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_QUEUE

The length of the outbound queue at the time of the last sample. This metric will be the same as the “Outbound Queue Length” values from the output of “lanadmin” utility.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On HP-UX, this metric is only available for LAN interfaces. For WAN (Wide-Area Network) interfaces such as ATM and X.25, with interface names such as el, cip/ixe, and netisdn, this metric returns “na”.

BYNETIF_UTIL

The percentage of bandwidth used with respect to the total available bandwidth on a given network interface at the end of the interval.

On vMA this value will be N/A for those Lan cards which are of type ESXVLan.

BYNFSOP_CLIENT_COUNT

The number of operations that the local machine processed as a client for the current host during the interval.

BYNFSOP_CLIENT_COUNT_CUM

The number of operations that the local machine processed as a client for the current host over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFSOP_CLIENT_TIME

The time, in seconds, spent to service an NFS operation (as an NFS client) during the last interval. This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the “service time” less the local machine's software overhead.

BYNFSOP_CLIENT_TIME_CUM

The time, in seconds, spent to service an NFS operation (as an NFS client) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the "service time" less the local machine's software overhead.

BYNFSOP_INTERVAL

The amount of time in the interval.

BYNFSOP_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFSOP_NAME

String mnemonic for the NFS operation. One of the following:

```
NFS Version 2

Name        Operation/Action
------------------------------------
getattr     Return the current
            attributes of a file.
setattr     Set the attributes of a
            file and returns the new
            attributes.
lookup      Return the attributes of
            a file.
readlink    Return the string in the
            symbolic link of a file.
read        Return data from a file.
```

```
write       Put data into a file.
create      Create a file.
remove      Remove a file.
rename      Give a file a new name.
link        Create a hard link to a
            file.
symlink     Create a symbolic link
            to a file.
mkdir       Create a directory.
rmdir       Remove a directory.
readdir     Read a directory entry.
statfs      Return mounted file
            system information.
null        Verify NFS services.
            No actual work done.
writecache  Not used in HP-UX.
root        Not used in HP-UX.

NFS Version 3

Name        Operation/Action
------------------------------------
getattr     Return the current
            attributes of a file.
setattr     Set the attributes of a
            file and returns the new
            attributes.
lookup      Return the attributes of
            a file.
access      Check access permissions
            of a user.
readlink    Return the string in the
            symbolic link of a file.
read        Return data from a file.
write       Put data into a file.
create      Create a file.
mkdir       Make a directory.
symlink     Create a symbolic link
            to a file.
mknod       Create a special device.
remove      Remove a file.
rmdir       Remove a directory.
rename      Give a file a new name.
link        Create a hard link to a
            file.
readdir     Read a directory entry.
readdirplus Extended read of a
            directory entry.
fsstat      Get dynamic file
            system information.
```

```
fsinfo       Get static file
             system information.
pathconf     Retrieve POSIX
             information.
commit       Commit cached data on
             server to stable
             storage.
null         Verify NFS services.
             No actual work done.
```

BYNFSOP_SERVER_COUNT

The number of NFS operations that the local machine performed as a server to the current host for this current operation type during the interval.

BYNFSOP_SERVER_COUNT_CUM

The number of NFS operations that the local machine performed as a server to the current host for this operation type over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFSOP_SERVER_TIME

The time, in seconds, that the local machine spent servicing each NFS operation as a NFS server for the current host during the interval. This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the “service time” less the local machine's software overhead.

BYNFSOP_SERVER_TIME_CUM

The time, in seconds, that the local machine spent servicing each NFS operation as a NFS server for the current host over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the “service time” less the local machine's software overhead.

BYNFS_CLIENT_PHYS_TIME

The time, in seconds, that the local machine spent to service all NFS operations (as an NFS client) to this host entry during the interval.

This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the “service time” less the local machine's software overhead.

BYNFS_CLIENT_PHYS_TIME_CUM

The time, in seconds, that the local machine spent to service all NFS operations (as a NFS client) to this host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the “service time” less the local machine's software overhead.

BYNFS_CLIENT_READ_BYTE_RATE

The number of KBs per second transferred during the interval by the NFS read operations where the local machine was acting as a client for this host.

BYNFS_CLIENT_READ_BYTE_RATE_CUM

The average number of KBs per second transferred by the NFS read operations where the local machine was acting as a client for this host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_CLIENT_READ_RATE

The number of NFS read operations per second where the local machine was acting as a client to this NFS host entry during the interval.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

BYNFS_CLIENT_READ_RATE_CUM

The average number of NFS read operations per second where the local machine was acting as a client to this NFS host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

BYNFS_CLIENT_SERVICE

The number of NFS IO operations processed by the local machine acting as a client for this host entry during the interval. This is sometimes referred to as the “service count.”

BYNFS_CLIENT_SERVICE_CUM

The number of NFS IO operations processed by the local machine acting as a client to this host entry over the cumulative collection time. This is sometimes referred to as the “service count.”

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_CLIENT_SERVICE_QUEUE

The local machine's number of pending NFS client read or write operations to this NFS host at the end of the interval. This value increases as the service time to the NFS host increases and/or as the rate of client requests increases.

A large value is an indication that either the NFS server is busy, or the local machine is a heavy user of the current server, or both.

BYNFS_CLIENT_SERVICE_QUEUE_CUM

The local machine's average number of pending NFS client read or write operations to this NFS host over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

The length of this queue increases as the service time to the NFS host increases and/or as the rate of the local machine's requests increases. A large value is an indication that either the NFS server is busy, or the local machine is a heavy user of the current server, or both.

BYNFS_CLIENT_SERVICE_TIME

The time, in seconds, spent for the local machine acting as an client to service all NFS operations for this host entry during the interval.

This is the time from the point that the local machine (as a client) originates the request to the point a reply is received including IO buffering, NFS and network software layer delays, physical network latency, and NFS server service time. This is sometimes referred to as “service time” and can be thought of as the round-trip time.

BYNFS_CLIENT_SERVICE_TIME_CUM

The time, in seconds, spent for the local machine acting as a NFS client for this host entry to service all NFS operations over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This is the time from the point that the local machine (as a client) originates the request to the point a reply is received including IO buffering, NFS and network software layer delays, physical network latency, and NFS server service time. This is sometimes referred to as “service time” and can be thought of as the round-trip time.

BYNFS_CLIENT_WRITE_BYTE_RATE

The number of KBs per second transferred by the NFS write operation where the local machine was acting as a client for this host entry during the interval.

BYNFS_CLIENT_WRITE_BYTE_RATE_CUM

The average number of KBs per second transferred by the NFS write operation where the local machine was acting as a client for this host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_CLIENT_WRITE_RATE

The number of NFS write operations per second where the local machine was acting as a client to this NFS host entry during the interval.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

BYNFS_CLIENT_WRITE_RATE_CUM

The average number of NFS write operations per second where the local machine was acting as a client to this NFS host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

BYNFS_HOSTNAME

The Internet host name of this NFS entry.

An NFS host is added if there are already NFS directories mounted or whenever any IO activity is seen, either server or client activity. It remains listed as long as the current midaemon program is running even if all NFS file systems are unmounted.

A host on the network can act both as a client, or as a server at the same time. If an NFS host acts as both client and as a server, it is only listed once.

BYNFS_HOST_IP_ADDRESS

The Internet host IP address of this NFS entry.

BYNFS_INTERVAL

The amount of time in the interval.

BYNFS_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_LAST_PROC_ID

The PID of the last process to generate or receive NFS traffic for this NFS host.

If the host is acting as a server (that is, the local machine is the client), then the last process may be either a user application or the biod daemon.

If the host entry is acting as a client (that is, the local machine is the server), then this process is always the nfsd daemon.

BYNFS_SERVER_READ_BYTE_RATE

The number of KBs per second transferred during the interval by the NFS read operations where the local machine was acting as a server to this host.

BYNFS_SERVER_READ_BYTE_RATE_CUM

The average number of KBs per second transferred by the NFS read operations where the local machine was acting as a server to this host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_SERVER_READ_RATE

The number of NFS read operations per second where the local machine was acting as a server for this NFS host entry during the interval.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

BYNFS_SERVER_READ_RATE_CUM

The average number of NFS read operations per second where the local machine was acting as a server for this NFS host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

BYNFS_SERVER_SERVICE

The number of NFS IO operations processed by the local machine acting as a server to this host entry during the interval. This is sometimes referred to as the “service count.”

BYNFS_SERVER_SERVICE_CUM

The number of NFS IO operations processed by the local machine acting as a server to this host entry over the cumulative collection time. This is sometimes referred to as the “service count.”

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_SERVER_SERVICE_TIME

The time, in seconds, spent for the local machine acting as a NFS server to this host entry to process the client's operations during the interval. This includes all of the time from the point that the operation is received to the point where a reply is sent back to the client, which includes software overhead and any local disk IOs.

BYNFS_SERVER_SERVICE_TIME_CUM

The time, in seconds, spent over the cumulative collection time for the local machine acting as a NFS server to this host entry to process the client's operations. This includes all of the time from the point that the operation is received to the point where a reply is sent back to the client, which includes software overhead and any local disk IOs.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_SERVER_WRITE_BYTE_RATE

The number of KBs per second transferred by the NFS write operation where the local machine was acting as a server to this host entry during the interval.

BYNFS_SERVER_WRITE_BYTE_RATE_CUM

The average number of KBs per second transferred by the NFS write operation where the local machine was acting as a server to this host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYNFS_SERVER_WRITE_RATE

The number of NFS write operations per second where the local machine was acting as a server for this NFS host entry during the interval.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

BYNFS_SERVER_WRITE_RATE_CUM

The average number of NFS write operations per second where the local machine was acting as a server for this NFS host entry over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

BYOP_CLIENT_COUNT

The number of current NFS operations that the local machine has processed as a NFS client during the interval.

A host on the network can act both as a client, or as a server at the same time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYOP_CLIENT_COUNT_CUM

The number of current NFS operations that the local machine has processed as a NFS client over the cumulative collection time.

A host on the network can act both as a client, or as a server at the same time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYOP_INTERVAL

The amount of time in the interval.

BYOP_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYOP_NAME

String mnemonic for the NFS operation. One of the following:

```
For NFS Version 2

Name        Operation/Action
-------------------------------------
getattr     Return the current
            attributes of a file.
setattr     Set the attributes of a
            file and returns the new
            attributes.
lookup      Return the attributes of
            a file.
readlink    Return the string in the
            symbolic link of a file.
read        Return data from a file.
write       Put data into a file.
create      Create a file.
remove      Remove a file.
rename      Give a file a new name.
link        Create a hard link to a
            file.
symlink     Create a symbolic link
            to a file.
mkdir       Create a directory.
rmdir       Remove a directory.
readdir     Read a directory entry.
statfs      Return mounted file
            system information.
null        Verify NFS service
            connections and timing.
            On HP-UX, no actual work
            done.
writecache  Flush the server write
            cache if a special write
            cache exists.  Most
            systems use the file
```

	buffer cache and not a special server cache. Not used on HP-UX.
root	Find root file system handle (probably obsolete). Not used on HP-UX.

For NFS Version 3

Name	Operation/Action
getattr	Return the current attributes of a file.
setattr	Set the attributes of a file and returns the new attributes.
lookup	Return the attributes of a file.
access	Check access permissions of a user.
readlink	Return the string in the symbolic link of a file.
read	Return data from a file.
write	Put data into a file.
create	Create a file.
mkdir	Make a directory.
symlink	Create a symbolic link to a file.
mknod	Create a special device.
remove	Remove a file.
rmdir	Remove a directory.
rename	Give a file a new name.
link	Create a hard link to a file.
readdir	Read a directory entry.
readdirplus	Extended read of a directory entry.
fsstat	Get dynamic file system information.
fsinfo	Get static file system information.
pathconf	Retrieve POSIX information.
commit	Commit cached data on server to stable storage.
null	Verify NFS services. No actual work done.

BYOP_SERVER_COUNT

The number of current NFS operations that the local machine has processed as a NFS server during the interval.

A host on the network can act both as a client, or as a server at the same time.

BYOP_SERVER_COUNT_CUM

The number of current NFS operations that the local machine has processed as a NFS server over the cumulative collection time.

A host on the network can act both as a client, or as a server at the same time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

BYSWP_SWAP_PRI

The priority of this swap device. This value is set by either the swapon(1M) command, or by the “pri=” field in /etc/fstab.

On HP-UX, swap space is used by the lower value priorities first. Since device swap is faster than file system swap, it is advisable to have lower values for device swap. The legal values for priority range from 0 to 10.

On HP-UX, the “memory” swap area has no priority and will be shown as -1. This indicates that using memory as a swap area is only done after all other swap resources have been exhausted. This is true in extreme cases of memory pressure forcing the kernel to swap the entire process to disk. In cases of process deactivation, the memory pseudo swap actually has the highest priority - deactivated pages are not moved - they are simply marked as deactivated and the space they occupy is considered pseudo swap.

On Linux, swap space is used by the higher value priorities first. The legal values for priority range from 0 to 32767. The system assigns negative priority values if no priority is specified during the creation of swap area. See swapon(8) for details.

BYSWP_SWAP_SPACE_AVAIL

The capacity (in MB) for swapping in this swap area.

On HP-UX, for “device” type swap, this value is constant. However, for “filesys” swap this value grows as needed. File system swap grows in units of “SWCHUNKS” x DEV_BSIZE bytes, which is typically 2MB. This metric is similar to the “AVAIL” parameters returned from /usr/sbin/swapinfo. For “memory” type swap, this value also grows as needed or as possible, given that any memory reserved for swap cannot be used for normal virtual memory. Note that this is potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space.

On SUN, this is the same as (blocks * .5)/1024, reported by the “swap -l” command.

On AIX, this metric is set to “na” for inactive swap devices.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

BYSWP_SWAP_SPACE_NAME

On Unix systems, this is the name of the device file or file system where the swap space is located.

On HP-UX, part of the system's physical memory may be allocated as a pseudo-swap device. It is enabled by setting the “SWAPMEM_ON” kernel parameter to 1.

On SunOS 5.X, part of the system's physical memory may be allocated as a pseudo-swap device. Also note, “/tmp” is usually configured as a memory based file system and is not used for swap space. Therefore, it will not be listed with the swap devices. This is noted because “df” uses the label “swap” for the “/tmp” file system which may be confusing. See tmpfs(7).

BYSWP_SWAP_SPACE_USED

The amount of swap space (in MB) used in this area.

On HP-UX, this value is similar to the “USED” column returned by the /usr/sbin/swapinfo command.

On SUN, “Used” indicates amount written to disk (or locked in memory), rather than reserved. Swap space is reserved (by decrementing a counter) when virtual memory for a program is created. This is the same as (blocks - free) * .5/1024, reported by the “swap -l” command.

On SUN, global swap space is tracked through the operating system. Device swap space is tracked through the devices. For this reason, the amount of swap space used may differ between the global and by-device metrics. Sometimes pages that are marked to be swapped to disk by the operating system are never swapped. The operating system records this as used swap space, but the devices do not, since no physical IOs occur. (Metrics with the prefix “GBL” are global and metrics with the prefix “BYSWP” are by device.)

On AIX, this metric is set to “na” for inactive swap devices.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

BYSWP_SWAP_TYPE

The type of swap space allocated on the system.

On HP-UX and SUN, types of swap space are device, file system ("filesys"), or memory. "Device" swap is accessed directly without going through the file system, and is therefore faster than "filesys" swap. "Filesys" swap can be to a local or NFS mounted swap file. "Memory" swap is space in the system's physical memory reserved for pseudo-swap for running processes. Using pseudo-swap means the pages are simply locked in memory rather than copied to a swap area.

On SUN, note that "/tmp" is usually configured as a memory based file system and is not used for swap space. Therefore, it will not be listed with the swap devices, and "swap" or "tmpfs" will not be swap types. This is noted because "df" uses the label "swap" for the "/tmp" file system which may be confusing. See tmpfs(7).

On AIX, "Device" swap is accessed directly without going through the file system. For "Device" swap, the device is specially allocated for swapping purpose only. The device can be logical volume, "lv" or remote file system, "remote fs". The swap is often referred as paging to paging space.

FSDETAIL_LABEL

The type of entry this file system is associated with. It could be a device, partition, logical volume, or volume group.

FSDETAIL_NAME

The name of the device, partition, logical volume, or volume group this file system is associated with.

FS_BLOCK_SIZE

The maximum block size of this file system, in bytes.

A value of "na" may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_DEVNAME

On Unix systems, this is the path name string of the current device.

On Windows, this is the disk drive string of the current device.

On HP-UX, this is the “fsname” parameter in the mount(1M) command. For NFS devices, this includes the name of the node exporting the file system. It is possible that a process may mount a device using the mount(2) system call. This call does not update the “/etc/mnttab” and its name is blank. This situation is rare, and should be corrected by syncer(1M). Note that once a device is mounted, its entry is displayed, even after the device is unmounted, until the midaemon process terminates.

On SUN, this is the path name string of the current device, or “tmpfs” for memory based file systems. See tmpfs(7).

FS_DEVNO

On Unix systems, this is the major and minor number of the file system.

On Windows, this is the unit number of the disk device on which the logical disk resides.

The scope collector logs the value of this metric in decimal format.

FS_DIRNAME

On Unix systems, this is the path name of the mount point of the file system.

On Windows, this is the drive letter associated with the selected disk partition.

On HP-UX, this is the path name of the mount point of the file system if the logical volume has a mounted file system. This is the directory parameter of the mount(1M) command for most entries. Exceptions are:

```
* For lvm swap areas, this field
  contains “lvm swap device”.
* For logical volumes with no
  mounted file systems, this field
  contains “Raw Logical Volume”
  (relevant only to Perf Agent).
```

On HP-UX, the file names are in the same order as shown in the “/usr/sbin/mount -p” command. File systems are not displayed until they exhibit IO activity once the midaemon has been started. Also, once a device is displayed, it continues to be displayed (even after the device is unmounted) until the midaemon process terminates.

On SUN, only “UFS”, “HSFS” and “TMPFS” file systems are listed. See mount(1M) and mnttab(4). “TMPFS” file systems are memory based filesystems and are listed here for convenience. See tmpfs(7).

On AIX, see mount(1M) and filesystems(4). On OSF1, see mount(2).

FS_FILE_IO_RATE

The number of file system related physical IOs per second directed to this file system during the interval.

This value is similar to the values returned by the vmstat -d command except that vmstat reports all IOs and does not break them out by file system. Also, vmstat reports IOs from the kernel's view, which may get broken down by the disk driver into multiple physical IOs. Since this metric reports values from the disk driver's point of view, it is more accurate than vmstat.

FS_FILE_IO_RATE_CUM

The average number of file IOs per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This value is similar to the values returned by the vmstat -d command except that vmstat reports all IOs and does not break them out by file system. Also, vmstat reports IOs from the kernel's view, which may get broken down by the disk driver into multiple physical IOs. Since this metric reports values from the disk driver's point of view, it is more accurate than vmstat.

FS_FRAG_SIZE

The fundamental file system block size, in bytes.

A value of “na” may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_INODE_UTIL

Percentage of this file system'sinodes in use during the interval.

A value of “na” may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_INTERVAL

The amount of time in the interval.

FS_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_IS_LVM

Returns true (1) if this file system is a logical volume or 0 if a hard-partitioned file system.

FS_LOGL_IO_RATE

The number of logical IOs per second directed to this file system during the interval. Logical IOs are generated by calling the read() or write() system calls.

FS_LOGL_IO_RATE_CUM

The average number of logical IOs per second directed to this file system over the cumulative collection time. Logical IOs are generated by calling the read() or write() system calls.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_LOGL_READ_BYTE_RATE

The number of logical read KBs per second from this file system during the interval.

FS_LOGL_READ_BYTE_RATE_CUM

The average number of logical read KBs per second from this file system over the cumulative collection time. Logical reads are generated by calling the read() system call.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_LOGL_READ_RATE

The number of logical reads per second directed to this file system during the interval. Logical reads are generated by calling the read() system call.

FS_LOGL_READ_RATE_CUM

The average number of logical reads per second directed to this file system over the cumulative collection time. Logical reads are generated by calling the read() system call.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_LOGL_WRITE_BYTE_RATE

The number of logical writes KBs per second to this file system during the interval.

FS_LOGL_WRITE_BYTE_RATE_CUM

The average number of logical write KBs per second to this file system over the cumulative collection time. Logical writes are generated by calling the write() system call.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_LOGL_WRITE_RATE

The number of logical writes per second directed to this file system during the interval. Logical writes are generated by calling the write() system call.

FS_LOGL_WRITE_RATE_CUM

The average number of logical writes per second directed to this file system over the cumulative collection time. Logical writes are generated by calling the write() system call.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_MAX_INODES

Number of configured file systeminodes.

A value of “na” may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_MAX_SIZE

Maximum number that this file system could obtain if full, in MB.

Note that this is the user space capacity - it is the file system space accessible to non root users. On most Unix systems, the df command shows the total file system capacity which includes the extra file system space accessible to root users only.

The equivalent fields to look at are “used” and “avail”. For the target file system, to calculate the maximum size in MB, use

```
FS Max Size = (used + avail)/1024
```

A value of “na” may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

On HP-UX, this metric is updated at 4 minute intervals to minimize collection overhead.

FS_PHYS_IO_RATE

The number of physical IOs per second directed to this file system during the interval.

FS_PHYS_IO_RATE_CUM

The average number of physical IOs per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_PHYS_READ_BYTE_RATE

The number of physical KBs per second read from this file system during the interval.

FS_PHYS_READ_BYTE_RATE_CUM

The average number of KBs per second of physical reads from this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_PHYS_READ_RATE

The number of physical reads per second directed to this file system during the interval.

On Unix systems, physical reads are generated by user file access, virtual memory access (paging), file system management, or raw device access.

FS_PHYS_READ_RATE_CUM

The average number of physical reads per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_PHYS_WRITE_BYTE_RATE

The number of physical KBs per second written to this file system during the interval.

FS_PHYS_WRITE_BYTE_RATE_CUM

The average number of KBs per second of physical writes to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_PHYS_WRITE_RATE

The number of physical writes per second directed to this file system during the interval.

FS_PHYS_WRITE_RATE_CUM

The average number of physical writes per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

FS_SPACE_RESERVED

The amount of file system space in MBs reserved for superuser allocation.

On AIX, this metric is typically zero because by default AIX does not reserve any file system space for the superuser.

FS_SPACE_USED

The amount of file system space in MBs that is being used.

FS_SPACE_UTIL

Percentage of the file system space in use during the interval.

Note that this is the user space capacity - it is the file system space accessible to non root users. On most Unix systems, the df command shows the total file system capacity which includes the extra file system space accessible to root users only.

A value of “na” may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

On HP-UX, this metric is updated at 4 minute intervals to minimize collection overhead.

FS_TYPE

A string indicating the file system type. On Unix systems, some of the possible types are:

```
hfs   - user file system
ufs   - user file system
ext2  - user file system
cdfs  - CD-ROM file system
vxfs  - Veritas (vxfs) file system
nfs   - network file system
nfs3  - network file system
        Version 3
```

On Windows, some of the possible types are:

```
NTFS  - New Technology File System
FAT   - 16-bit File Allocation
        Table
FAT32 - 32-bit File Allocation
        Table
```

FAT uses a 16-bit file allocation table entry (216 clusters).

FAT32 uses a 32-bit file allocation table entry. However, Windows 2000 reserves the first 4 bits of a FAT32 file allocation table entry, which means FAT32 has a theoretical maximum of 228 clusters. NTFS is native file system of Windows NT and beyond.

FS_VM_IO_RATE

The number of virtual memory IOs per second directed to this file system during the interval.

FS_VM_IO_RATE_CUM

The average number of virtual memory IOs per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_ACTIVE_CPU

The number of CPUs online on the system.

For HP-UX and certain versions of Linux, the sar(1M) command allows you to check the status of the system CPUs.

For SUN and DEC, the commands psrinfo(1M) and psradm(1M) allow you to check or change the status of the system CPUs.

For AIX, the pstat(1) command allows you to check the status of the system CPUs.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment if RSET is not configured for the System WPAR. If RSET is configured for the System WPAR, this metric value will report the number of CPUs in the RSET.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_ACTIVE_CPU_CORE

This metric provides the total number of active CPU cores on a physical system.

GBL_ACTIVE_PROC

An active process is one that exists and consumes some CPU time. GBL_ACTIVE_PROC is the sum of the alive-process-time/interval-time ratios of every process that is active (uses any CPU time) during an interval.

The following diagram of a four second interval during which two processes exist on the system should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

```
     ----------- Seconds -----------
       1          2          3      4
Proc
---- ----       ----       ----   ----
A    live       live       live   live

B    live/CPU   live/CPU   live   dead
```

Process A is alive for the entire four second interval but consumes no CPU. A's contribution to GBL_ALIVE_PROC is 4*1/4. A contributes 0*1/4 to GBL_ACTIVE_PROC. B's contribution to GBL_ALIVE_PROC is 3*1/4. B contributes 2*1/4 to GBL_ACTIVE_PROC. Thus, for this interval, GBL_ACTIVE_PROC equals 0.5 and GBL_ALIVE_PROC equals 1.75.

Because a process may be alive but not active, GBL_ACTIVE_PROC will always be less than or equal to GBL_ALIVE_PROC.

This metric is a good overall indicator of the workload of the system. An unusually large number of active processes could indicate a CPU bottleneck.

To determine if the CPU is a bottleneck, compare this metric with GBL_CPU_TOTAL_UTIL and GBL_RUN_QUEUE. If GBL_CPU_TOTAL_UTIL is near 100 percent and GBL_RUN_QUEUE is greater than one, there is a bottleneck.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

GBL_ALIVE_PROC

An alive process is one that exists on the system. GBL_ALIVE_PROC is the sum of the alive-process-time/interval-time ratios for every process.

The following diagram of a four second interval during which two processes exist on the system should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

```
     ----------- Seconds -----------
       1          2          3      4
Proc
---- ----       ----       ----   ----
A    live       live       live   live
```

```
B    live/CPU  live/CPU  live   dead
```

Process A is alive for the entire four second interval but consumes no CPU. A's contribution to GBL_ALIVE_PROC is 4*1/4. A contributes 0*1/4 to GBL_ACTIVE_PROC. B's contribution to GBL_ALIVE_PROC is 3*1/4. B contributes 2*1/4 to GBL_ACTIVE_PROC. Thus, for this interval, GBL_ACTIVE_PROC equals 0.5 and GBL_ALIVE_PROC equals 1.75.

Because a process may be alive but not active, GBL_ACTIVE_PROC will always be less than or equal to GBL_ALIVE_PROC.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

GBL_BLANK

A string of blanks.

GBL_BOOT_TIME

The date and time when the system was last booted.

GBL_CACHE_QUEUE

The average number of processes or kernel threads blocked on CACHE (waiting for the file systembuffer cache to be updated) during the interval. Processes or kernel threads doing raw IO to a disk are not included in this measurement. As this number rises, it is an indication of a disk or memory bottleneck.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on CACHE divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being

examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_CACHE_WAIT_PCT

The percentage of time processes or kernel threads were blocked on cache (waiting for the file systembuffer cache to be updated) during the interval. Processes or kernel threads doing raw IO to a disk are not included in this measurement.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on CACHE divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_CACHE_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on CACHE (waiting for the file systembuffer cache to be updated) during the interval. Processes or kernel threads doing raw IO to a disk are not included in this measurement.

GBL_CDFS_QUEUE

The average number of processes or kernel threads blocked on CDFS (waiting for their Compact Disk file system IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on CDFS divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_CDFS_WAIT_PCT

The percentage of time processes or kernel threads were blocked on CDFS (waiting for their Compact Disk file system IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on CDFS divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_CDFS_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on CDFS (waiting for their Compact Disk file system IO to complete) during the interval.

GBL_COLLECTOR

ASCII field containing collector name and version. The collector name will appear as either “SCOPE/xx V.UU.FF.LF” or “Coda RV.UU.FF.LF”. xx identifies the platform; V = version, UU = update level, FF = fix level, and LF = lab fix id. For example, SCOPE/UX C.04.00.00; or Coda A.07.10.04.

GBL_COMPLETED_PROC

The number of processes that terminated during the interval.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

GBL_CPU_CLOCK

The clock speed of the CPUs in MHz if all of the processors have the same clock speed. Otherwise, “na” is shown if the processors have different clock speeds. Note that Linux supports dynamic frequency scaling and if it is enabled then there can be a change in CPU speed with varying load.

GBL_CPU_CSWITCH_TIME

The time, in seconds, that the CPU spent context switching during the interval.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_CSWITCH_TIME_CUM

The time, in seconds, that the CPU spent context switching over the cumulative collection time. On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_CSWITCH_UTIL

The percentage of time that the CPU spent context switching during the interval.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_CSWITCH_UTIL_CUM

The percentage of time that the CPU spent context switching over the cumulative collection time. On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_CSWITCH_UTIL_HIGH

The highest percentage of time during any one interval that the CPU spent context switching over the cumulative collection time. On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_TIME

The time, in seconds, that the CPU was idle during the interval. This is the total idle time, including waiting for I/O.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online.

On AIX System WPARs, this metric value is calculated against physical cpu time.

On Solaris non-global zones, this metric is N/A. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_TIME_CUM

The time, in seconds, that the CPU was idle over the cumulative collection time. This is the total idle time, including waiting for I/O.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_UTIL

The percentage of time that the CPU was idle during the interval. This is the total idle time, including waiting for I/O.

On Unix systems, this is the same as the sum of the "%idle" and "%wio" fields reported by the "sar -u" command.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online.

On Solaris non-global zones, this metric is N/A. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_UTIL_CUM

The percentage of time that the CPU was idle over the cumulative collection time. This is the total idle time, including waiting for I/O.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_UTIL_HIGH

The highest percentage of time that the CPU was idle during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_TIME

The time, in seconds, that the CPU spent processing interrupts during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On Hyper-V host, this metric is NA.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_TIME_CUM

The time, in seconds, that the CPU spent processing interrupts over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_UTIL

The percentage of time that the CPU spent processing interrupts during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On Hyper-V host, this metric is NA.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be

added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_UTIL_CUM

The percentage of time that the CPU spent processing interrupts over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_UTIL_HIGH

The highest percentage of time that the CPU spent processing interrupts during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_MT_ENABLED

On AIX, this metric indicates if this (Logical) System has SMT enabled or not.

Other platforms, this metric shows either HyperThreading(HT) is Enabled or Disabled/Not Supported.

On Linux, this state is dynamic: if HyperThreading is enabled but all the CPUs have only one logical processor enabled, this metric will report that HT is disabled.

On AIX System WPARs, this metric is NA.

On Windows, this metric will be “na” on Windows Server 2003 Itanium systems.

GBL_CPU_NICE_TIME

The time, in seconds, that the CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_TIME_CUM

The time, in seconds, that the CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_UTIL

The percentage of time that the CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_UTIL_CUM

The percentage of time that the CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_UTIL_HIGH

The highest percentage of time during any one interval that the CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NNICE_TIME

The time, in seconds, that the CPU was in user mode at a nice priority calculated from processes with negative nice values during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total

processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NNICE_TIME_CUM

The time, in seconds, that the CPU was in user mode at a nice priority calculated from processes with negative nice values over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NNICE_UTIL

The percentage of time that the CPU was in user mode at a nice priority calculated from processes with negative nice values during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NNICE_UTIL_CUM

The percentage of time that the CPU was in user mode at a nice priority calculated from processes with negative nice values over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NNICE_UTIL_HIGH

The highest percentage of time during any one interval that the CPU was in user mode at a nice priority calculated from processes with negative nice values over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NORMAL_TIME

The time, in seconds, that the CPU was in user mode at normal priority during the interval. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NORMAL_TIME_CUM

The time, in seconds, that the CPU was in user mode at normal priority over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NORMAL_UTIL

The percentage of time that the CPU was in user mode at normal priority during the interval. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NORMAL_UTIL_CUM

The percentage of time that the CPU was in user mode at normal priority over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NORMAL_UTIL_HIGH

The highest percentage of time that the CPU was in user mode at normal priority during any one interval over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_QUEUE

The average number of processes or kernel threads using the CPU plus all of those processes or kernel threads blocked on PRIORITY (waiting for their priority to become high enough to get the CPU) during the interval. This metric is an indicator of CPU demands among the active processes or kernel threads.

To determine if the CPU is a bottleneck, compare this metric with GBL_CPU_TOTAL_UTIL. If GBL_CPU_TOTAL_UTIL is near 100 percent and GBL_CPU_QUEUE is greater than four, there is a high probability of a CPU bottleneck.

This is calculated as (the CPU time used plus the accumulated time that all processes or kernel threads spent blocked on PRI (that is, priority)) divided by the interval time.

The difference between this metric and GBL_PRI_QUEUE is that it includes the processes or kernel threads using the CPU, if any.

HP-UX RUN/PRI/CPU Queue differences for multi-cpu systems:

For example, let's assume we're using a system with eight processors. We start eight CPU intensive threads that consume almost all of the CPU resources. The approximate values shown for the CPU related queue metrics would be:

```
GBL_RUN_QUEUE = 1.0
GBL_PRI_QUEUE = 0.1
GBL_CPU_QUEUE = 1.0
```

Assume we start an additional eight CPU intensive threads. The approximate values now shown are:

```
GBL_RUN_QUEUE = 2.0
GBL_PRI_QUEUE = 8.0
GBL_CPU_QUEUE = 16.0
```

At this point, we have sixteen CPU intensive threads running on the eight processors. Keeping the definitions of the three queue metrics in mind, the run queue is 2 (that is, 16 / 8); the pri queue is 8 (only half of the threads can be active at any given time); and the cpu queue is 16 (half of the threads waiting in the cpu queue that are ready to run, plus one for each active thread).

This illustrates that the run queue is the average of number of threads waiting in the runqueue for all processors; the pri queue is the number of threads that are blocked on “PRI” (priority); and the cpu queue is the number of threads in the cpu queue that are ready to run, including the threads using the CPU.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being

examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_CPU_REALTIME_TIME

The time, in seconds, that the CPU was in user mode at a realtime priority during the interval. Running at a realtime priority means that the process or kernel thread was run using the rtprio command or the rtprio system call to alter its priority. Realtime priorities range from zero to 127 and are absolute priorities, meaning the realtime process with the lowest priority runs as long as it wants to. Since this can have a huge impact on the system, the realtime CPU is tracked separately to make visible the effect of using realtime priorities.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_REALTIME_TIME_CUM

The time, in seconds, that the CPU was in user mode at a realtime priority over the cumulative collection time. Running at a realtime priority means that the process or kernel thread was run using the rtprio command or the rtprio system call to alter its priority. Realtime priorities range from zero to 127 and are absolute priorities, meaning the realtime process with the lowest priority runs as long as it wants to. Since this can have a huge impact on the system, the realtime CPU is tracked separately to make visible the effect of using realtime priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_REALTIME_UTIL

The percentage of time that the CPU was in user mode at a realtime priority during the interval. Running at a realtime priority means that the process or kernel thread was run using the rtprio command or the rtprio system call to alter its priority. Realtime priorities range from zero to 127 and are absolute priorities, meaning the realtime process with the lowest priority runs as long as it wants to. Since this can have a huge impact on the system, the realtime CPU is tracked separately to make visible the effect of using realtime priorities.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_REALTIME_UTIL_CUM

The percentage of time that the CPU was in user mode at a realtime priority over the cumulative collection time. Running at a realtime priority means that the process or kernel thread was run using the rtprio command or the rtprio system call to alter its priority. Realtime priorities range from zero to 127 and are absolute priorities, meaning the realtime process with the lowest priority runs as long as it wants to. Since this can have a huge impact on the system, the realtime CPU is tracked separately to make visible the effect of using realtime priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_REALTIME_UTIL_HIGH

The highest percentage of time that the CPU was in user mode at a realtime priority during any one interval over the cumulative collection time. Running at a realtime priority means that the process or kernel thread was run using the rtprio command or the rtprio system call to alter its priority. Realtime priorities range from zero to 127 and are absolute priorities, meaning the realtime process with the lowest priority runs as long as it wants to. Since this can have a huge impact on the system, the realtime CPU is tracked separately to make visible the effect of using realtime priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_SYSCALL_TIME

The time, in seconds, that the CPU was in system mode (excluding interrupt, context switch, trap, or vfault CPU) during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_SYSCALL_TIME_CUM

The time, in seconds, that the CPU was in system mode (excluding interrupt, context switch, trap, or vfault CPU) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_SYSCALL_UTIL

The percentage of time that the CPU was in system mode (excluding interrupt, context switch, trap, or vfault CPU) during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_SYSCALL_UTIL_CUM

The percentage of time that the CPU was in system mode (excluding interrupt, context switch, trap, or vfault CPU) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total

processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_SYSCALL_UTIL_HIGH

The highest percentage of time that the CPU was in system mode (excluding interrupt, context switch, trap, or vfault CPU) during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be

added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_SYS_MODE_TIME

The time, in seconds, that the CPU was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

On Hyper-V host, this metric indicates the time spent in Hypervisor code.

GBL_CPU_SYS_MODE_TIME_CUM

The time, in seconds, that the CPU was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_SYS_MODE_UTIL

Percentage of time the CPU was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

This metric is a subset of the GBL_CPU_TOTAL_UTIL percentage.

This is NOT a measure of the amount of time used by system daemon processes, since most system daemons spend part of their time in user mode and part in system calls, like any other process.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

High system mode CPU percentages are normal for IO intensive applications. Abnormally high system mode CPU percentages can indicate that a hardware problem is causing a high interrupt rate. It can also indicate programs that are not calling system calls efficiently. On a logical system, this metric indicates the percentage of time the logical processor was in kernel mode during this interval.

On Hyper-V host, this metric indicates the percentage of time spent in Hypervisor code.

GBL_CPU_SYS_MODE_UTIL_CUM

The percentage of time that the CPU was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TOTAL_TIME

The total time, in seconds, that the CPU was not idle in the interval.

This is calculated as

```
GBL_CPU_TOTAL_TIME =
  GBL_CPU_USER_MODE_TIME +
  GBL_CPU_SYS_MODE_TIME
```

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_TOTAL_TIME_CUM

The total time that the CPU was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_TOTAL_UTIL

Percentage of time the CPU was not idle during the interval.

This is calculated as

```
GBL_CPU_TOTAL_UTIL =
  GBL_CPU_USER_MODE_UTIL +
  GBL_CPU_SYS_MODE_UTIL
```

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

```
  GBL_CPU_TOTAL_UTIL +
   GBL_CPU_IDLE_UTIL = 100%
```

This metric varies widely on most systems, depending on the workload. A consistently high CPU utilization can indicate a CPU bottleneck, especially when other indicators such as GBL_RUN_QUEUE and GBL_ACTIVE_PROC are also high. High CPU utilization can also occur on systems that are bottlenecked on memory, because the CPU spends more time paging and swapping.

NOTE: On Windows, this metric may not equal the sum of the APP_CPU_TOTAL_UTIL metrics. Microsoft states that "this is expected behavior" because this GBL_CPU_TOTAL_UTIL metric is taken from the performance library Processor objects while the APP_CPU_TOTAL_UTIL metrics are taken from the Process objects. Microsoft states that there can be CPU time accounted for in the Processor system objects that may not be seen in the Process objects. On a logical system, this metric indicates the logical utilization with respect to number of processors available for the logical system (GBL_NUM_CPU).

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TOTAL_UTIL_CUM

The percentage of total CPU time that the processor was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TOTAL_UTIL_HIGH

The highest percentage of total CPU time during any one interval that the processor was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TRAP_TIME

The time the CPU was in trap handler code during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TRAP_TIME_CUM

The time, in seconds, the CPU was in trap handler code over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TRAP_UTIL

The percentage of time the CPU was executing trap handler code during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TRAP_UTIL_CUM

The percentage of time the CPU was in trap handler code over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TRAP_UTIL_HIGH

The highest percentage of time during any one interval the CPU was in trap handler code over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_USER_MODE_TIME

The time, in seconds, that the CPU was in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup.

Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

On Hyper-V host, this metric indicates the time spent in guest code.

GBL_CPU_USER_MODE_TIME_CUM

The time, in seconds, that the CPU was in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_USER_MODE_UTIL

The percentage of time the CPU was in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

This metric is a subset of the GBL_CPU_TOTAL_UTIL percentage.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

High user mode CPU percentages are normal for computation-intensive applications. Low values of user CPU utilization compared to relatively high values for GBL_CPU_SYS_MODE_UTIL can indicate an application or hardware problem. On a logical system, this metric indicates the percentage of time the logical processor was in user mode during this interval.

On Hyper-V host, this metric indicates the percentage of time spent in guest code.

GBL_CPU_USER_MODE_UTIL_CUM

The percentage of time that the CPU was in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_VFAULT_TIME

The time, in seconds, the CPU was handling page faults during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_VFAULT_TIME_CUM

The time, in seconds, the CPU was handling page faults over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_VFAULT_UTIL

The percentage of time the CPU was handling page faults during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_VFAULT_UTIL_CUM

The percentage of time the CPU was handling page faults over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_VFAULT_UTIL_HIGH

The highest percentage of time during any one interval the CPU was handling page faults over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_WAIT_UTIL

The percentage of time during the interval that the CPU was idle and there were processes waiting for physical IOs to complete.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On Solaris non-global zones, this metric is N/A. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On Linux, this includes CPU steal time (shown as '%steal' in 'sar' and 'st' in 'vmstat').

GBL_CSWITCH_RATE

The average number of context switches per second during the interval.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

On Windows, this includes switches from one thread to another either inside a single process or across processes. A thread switch can be caused either by one thread asking another for information or by a thread being preempted by another higher priority thread becoming ready to run.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_CSWITCH_RATE_CUM

The average number of context switches per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

GBL_CSWITCH_RATE_HIGH

The highest number of context switches per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

GBL_DISK_FS_BYTE

The number of file system KBs (or MBs if specified) physically transferred to or from the disk during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are bytes transferred by user file system access and do not include bytes transferred via virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their bytes transferred in this category. They appear under virtual memory bytes transferred.

GBL_DISK_FS_BYTE_CUM

The number of file system KBs (or MBs if specified) transferred to or from the disk over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are bytes transferred by user file system access and do not include bytes transferred via virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their bytes transferred in this category. They appear under virtual memory bytes transferred.

GBL_DISK_FS_IO

The total of physical file system disk reads and writes during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

GBL_DISK_FS_IO_CUM

The total of file system disk physical reads and writes over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

GBL_DISK_FS_IO_PCT

The percentage of file system generated physical IOs of the total physical IOs during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

GBL_DISK_FS_IO_PCT_CUM

The percentage of file system generated physical IOs of the total physical IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

GBL_DISK_FS_IO_RATE

The total of file system disk physical reads and writes per second during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

GBL_DISK_FS_IO_RATE_CUM

The number of file system physical disk reads and writes per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

GBL_DISK_FS_READ

The number of file system disk reads during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical reads generated by user file system access and do not include virtual memory reads, system reads (inode access), or reads relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical reads in this category. They appear under virtual memory reads.

GBL_DISK_FS_READ_RATE

The number of file system disk reads per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical reads generated by user file system access and do not include virtual memory reads, system reads (inode access), or reads relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical reads in this category. They appear under virtual memory reads.

GBL_DISK_FS_WRITE

The number of file system disk writes during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical writes generated by user file system access and do not include virtual memory writes, system writes (inode updates), or writes relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical writes in this category. They appear under virtual memory writes.

GBL_DISK_FS_WRITE_RATE

The number of file system disk writes per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical writes generated by user file system access and do not include virtual memory writes, system writes (inode updates), or writes relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical writes in this category. They appear under virtual memory writes.

GBL_DISK_LOGL_BYTE_RATE

The number of KBs transferred per second via disk IO calls during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

GBL_DISK_LOGL_IO

The number of logical IOs made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_IO_CUM

The number of logical IOs made over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_IO_RATE

The number of logical IOs per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_IO_RATE_CUM

The average number of logical IOs per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_READ

On most systems, this is the number of logical reads made during the interval. On SUN, this is the number of logical block reads made during the interval. On Windows, this includes both buffered (cached) read requests and unbuffered reads.

Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_READ_BYTE

The number of KBs transferred through logical reads during the last interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

GBL_DISK_LOGL_READ_BYTE_CUM

The number of KBs transferred through logical reads over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

GBL_DISK_LOGL_READ_BYTE_RATE

The number of KBs transferred per second via logical reads during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

GBL_DISK_LOGL_READ_CUM

On most systems, this is the total number of logical reads made over the cumulative collection time. On SUN, this is the total number of logical block reads over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_READ_PCT

On most systems, this is the percentage of logical reads of the total logical IO during the interval. On SUN, this is the percentage of logical block reads of the total logical IOs during the interval.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_READ_PCT_CUM

On most systems, this is the percentage of logical reads of the total logical IOs over the cumulative collection time. On SUN, this is the percentage of logical block reads of the total logical IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_READ_RATE

On most systems, this is The average number of logical reads per second made during the interval. On SUN, this is the average number of logical block reads per second made during the interval. On Windows, this includes both buffered (cached) read requests and unbuffered reads.

Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_LOGL_READ_RATE_CUM

On most Unix systems, this is the average number of logical reads per second over the cumulative collection time. On SUN, this is the average number of logical block reads per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_WRITE

On most systems, this is the number of logical writes made during the interval. On SUN, this is the number of logical block writes during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_WRITE_BYTE

The number of KBs transferred via logical writes during the last interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

GBL_DISK_LOGL_WRITE_BYTE_CUM

The number of KBs transferred via logical writes over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

GBL_DISK_LOGL_WRITE_BYTE_RATE

The number of KBs per second transferred via logical writes during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

GBL_DISK_LOGL_WRITE_CUM

On most systems, this is the total number of logical writes made over the cumulative collection time. On SUN, this is the total number of logical block writes over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_WRITE_PCT

On most systems, this is the percentage of logical writes of the logical IO during the interval. On SUN, this is the percentage of logical block writes of the total logical block IOs during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_WRITE_PCT_CUM

On most systems, this is the percentage of logical writes of the total logical IO over the cumulative collection time. On SUN, this is the percentage of logical block writes of the total logical block IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_LOGL_WRITE_RATE

On most systems, this is the average number of logical writes per second made during the interval. On SUN, this is the average number of logical block writes per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_LOGL_WRITE_RATE_CUM

On most systems, this is the average number of logical writes per second of the total logical IOs over the cumulative collection time. On SUN, this is the average number of logical block writes per second of the total logical block IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

GBL_DISK_PHYS_BYTE

The number of KBs transferred to and from disks during the interval. The bytes for all types of physical IOs are counted. Only local disks are counted in this measurement. NFS devices are excluded.

It is not directly related to the number of IOs, since IO requests can be of differing lengths.

On Unix systems, this includes file system IO, virtual memory IO, and raw IO.

On Windows, all types of physical IOs are counted.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_BYTE_RATE

The average number of KBs per second at which data was transferred to and from disks during the interval. The bytes for all types physical IOs are counted. Only local disks are counted in this measurement. NFS devices are excluded.

This is a measure of the physical data transfer rate. It is not directly related to the number of IOs, since IO requests can be of differing lengths.

This is an indicator of how much data is being transferred to and from disk devices. Large spikes in this metric can indicate a disk bottleneck.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw reads.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_IO

The number of physical IOs during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk IOs are counted, including file system IO, virtual memory IO and raw IO.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_IO =
  GBL_DISK_FS_IO +
  GBL_DISK_VM_IO +
  GBL_DISK_SYSTEM_IO +
  GBL_DISK_RAW_IO
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_IO_CUM

The total number of physical IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_IO_RATE

The number of physical IOs per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk IOs are counted, including file system IO, virtual memory IO and raw IO.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_IO_RATE =
  GBL_DISK_FS_IO_RATE +
  GBL_DISK_VM_IO_RATE +
  GBL_DISK_SYSTEM_IO_RATE +
  GBL_DISK_RAW_IO_RATE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_IO_RATE_CUM

The number of physical IOs per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ

The number of physical reads during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On HP-UX, there are many reasons why there is not a direct correlation between the number of logical IOs and physical IOs. For example, small sequential logical reads may be satisfied from the buffer cache, resulting in fewer physical IOs than logical IOs. Conversely, large logical IOs or small random IOs may result in more physical than logical IOs. Logical volume mappings, logical disk mirroring, and disk striping also tend to remove any correlation.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_READ =
  GBL_DISK_FS_READ +
  GBL_DISK_VM_READ +
  GBL_DISK_SYSTEM_READ +
  GBL_DISK_RAW_READ
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_BYTE

The number of KBs physically transferred from the disk during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_BYTE_CUM

The number of KBs (or MBs if specified) physically transferred from the disk over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_BYTE_RATE

The average number of KBs transferred from the disk per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_CUM

The total number of physical reads over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_PCT

The percentage of physical reads of total physical IO during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_PCT_CUM

The percentage of physical reads of total physical IO over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_RATE

The number of physical reads per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_READ_RATE =
  GBL_DISK_FS_READ_RATE +
  GBL_DISK_VM_READ_RATE +
  GBL_DISK_SYSTEM_READ_RATE +
  GBL_DISK_RAW_READ_RATE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_RATE_CUM

The average number of physical reads per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE

The number of physical writes during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On HP-UX, there are many reasons why there is not a direct correlation between logical IOs and physical IOs. For example, small logical writes may end up entirely in the buffer cache, and later generate fewer physical IOs when written to disk due to the larger IO size. Or conversely, small logical writes may require physical prefetching of the corresponding disk blocks before the data is merged and posted to disk. Logical volume mappings, logical disk mirroring, and disk striping also tend to remove any correlation.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_WRITE =
  GBL_DISK_FS_WRITE +
  GBL_DISK_VM_WRITE +
  GBL_DISK_SYSTEM_WRITE +
  GBL_DISK_RAW_WRITE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_WRITE_BYTE

The number of KBs (or MBs if specified) physically transferred to the disk during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_BYTE_CUM

The number of KBs (or MBs if specified) physically transferred to the disk over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_BYTE_RATE

The average number of KBs transferred to the disk per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_WRITE_CUM

The total number of physical writes over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_PCT

The percentage of physical writes of total physical IO during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

GBL_DISK_PHYS_WRITE_PCT_CUM

The percentage of physical writes of total physical IO over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_RATE

The number of physical writes per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On HP-UX, this is calculated as

```
  GBL_DISK_PHYS_WRITE_RATE =
    GBL_DISK_FS_WRITE_RATE +
    GBL_DISK_VM_WRITE_RATE +
    GBL_DISK_SYSTEM_WRITE_RATE +
    GBL_DISK_RAW_WRITE_RATE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_WRITE_RATE_CUM

The number of physical writes per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_QUEUE

The average number of processes or kernel threads blocked on disk (in a “queue” within the disk drivers waiting for their file system disk IO to complete) during the interval. Processes or kernel threads doing raw IO to a disk are not included in this measurement. As this number rises, it is an indication of a disk bottleneck.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on DISK divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_DISK_RAW_BYTE

The number of KBs (or MBs if specified) transferred to or from a raw disk during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_RAW_BYTE_CUM

The number of KBs (or MBs if specified) transferred to or from a raw disk over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_RAW_IO

The total number of raw reads and writes during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Sun, tape drive accesses are included in raw IOs, but not in physical IOs. To determine if raw IO is tape access versus disk access, compare the global physical disk accesses to the total raw, block, and vm IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_RAW_IO_CUM

The total number of raw IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Sun, tape drive accesses are included in raw IOs, but not in physical IOs. To determine if raw IO is tape access versus disk access, compare the global physical disk accesses to the total raw, block, and vm IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_RAW_IO_PCT

The percentage of raw IOs to total physical IOs made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Sun, tape drive accesses are included in raw IOs, but not in physical IOs. To determine if raw IO is tape access versus disk access, compare the global physical disk accesses to the total raw, block, and vm IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_RAW_IO_PCT_CUM

The percentage of physical raw IOs to total physical IOs made over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Sun, tape drive accesses are included in raw IOs, but not in physical IOs. To determine if raw IO is tape access versus disk access, compare the global physical disk accesses to the total raw, block, and vm IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_RAW_IO_RATE

The total number of raw reads and writes per second during the interval. Only accesses to local disk devices are counted.

On Sun, tape drive accesses are included in raw IOs, but not in physical IOs. To determine if raw IO is tape access versus disk access, compare the global physical disk accesses to the total raw, block, and vm IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_RAW_IO_RATE_CUM

The average number of raw IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Sun, tape drive accesses are included in raw IOs, but not in physical IOs. To determine if raw IO is tape access versus disk access, compare the global physical disk accesses to the total raw, block, and vm IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_RAW_READ

The number of raw reads during the interval. Only accesses to local disk devices are counted.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_RAW_READ_RATE

The number of raw reads per second during the interval. Only accesses to local disk devices are counted.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_RAW_WRITE

The number of raw writes during the interval. Only accesses to local disk devices are counted.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_RAW_WRITE_RATE

The number of raw writes per second during the interval. Only accesses to local disk devices are counted.

On Sun, tape drive accesses are included in raw IOs, but not in physical IOs. To determine if raw IO is tape access versus disk access, compare the global physical disk accesses to the total raw, block, and vm IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_DISK_REM_FS_BYTE

The number of remote file system KBs (or MBs if specified) physically transferred to or from the remote machine during the interval.

These are bytes transferred by user file system access and do not include bytes transferred via virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their bytes transferred in this category. They appear under virtual memory bytes transferred.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_FS_BYTE_CUM

The number of remote file system KBs (or MBs if specified) transferred to or from the remote machine over the cumulative collection time.

These are bytes transferred by user file system access and do not include bytes transferred via virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their bytes transferred in this category. They appear under virtual memory bytes transferred.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_FS_IO

The total of remote physical file system reads and writes during the last interval.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_FS_IO_CUM

The total of remote file systemphysical reads and writes over the cumulative collection time.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_FS_IO_PCT

The percentage of remote file system generated physical IOs of the total remote physical IOs during the interval.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_FS_IO_PCT_CUM

The percentage of remote file system generated physical IOs of the total remote physical IOs over the cumulative collection time.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_FS_IO_RATE

The total of remote file systemphysical reads and writes per second during the interval.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_FS_IO_RATE_CUM

The total of remote file system physical reads and writes per second over the cumulative collection time.

These are physical IOs generated by user file system access and do not include virtual memory IOs, system IOs (inode updates), or IOs relating to raw disk access. An exception is user files accessed via the mmap(2) call, which will not show their physical IOs in this category. They appear under virtual memory IOs.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_LOGL_READ

The number of remote logical reads made during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_READ_BYTE

The number of KBs transferred via remote logical reads during the last interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_READ_BYTE_CUM

The number of KBs transferred via remote logical reads over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_READ_CUM

The total number of remote logical reads made over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_READ_PCT

The percentage of remote logical reads to the total remote logical IO during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_READ_PCT_CUM

The percentage of remote logical reads of the total remote logical IO over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_READ_RATE

The average number of remote logical reads per second made during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_READ_RATE_CUM

The average number of remote logical reads per second made over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE

The number of remote logical writes made during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE_BYTE

The number of KBs transferred via remote logical writes during the last interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE_BYTE_CUM

The number of KBs transferred via remote logical writes over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE_CUM

The total number of remote logical writes made over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE_PCT

The percentage of remote logical writes of the total remote logical IO during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE_PCT_CUM

The percentage of remote logical writes of the total remote logical IO over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE_RATE

The average number of remote logical writes per second made during the last interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_LOGL_WRITE_RATE_CUM

The percentage of remote logical writes of the total remote logical IO over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

GBL_DISK_REM_PHYS_READ

The number of remote physical reads during the interval. This includes all types of physical reads, including VM and raw.

This is calculated as

```
GBL_DISK_REM_PHYS_READ =
  GBL_DISK_REM_FS_READ +
  GBL_DISK_REM_VM_READ +
  GBL_DISK_REM_SYSTEM_READ +
  GBL_DISK_REM_RAW_READ
```

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_READ_BYTE

The number of physical read KBs during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_READ_BYTE_CUM

The number of physical read KBs (or MBs if specified) since collection was started or over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_READ_CUM

The total number of remote physical reads over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_READ_PCT

The percentage of remote physical reads of total remote physical IO during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_READ_PCT_CUM

The percentage of remote physical reads of total remote physical IO over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_READ_RATE

The number of remote physical reads per second during the interval. This includes all types of physical reads, including VM and raw.

This is calculated as

```
GBL_DISK_REM_PHYS_READ_RATE =
  GBL_DISK_REM_FS_READ_RATE +
  GBL_DISK_REM_VM_READ_RATE +
  GBL_DISK_REM_SYSTEM_READ_RATE +
  GBL_DISK_REM_RAW_READ_RATE
```

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_READ_RATE_CUM

The average number of remote physical reads per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE

The number of physical writes during the interval. All types of remote physical writes are counted, including VM and raw, are counted.

This is calculated as

```
GBL_DISK_REM_PHYS_WRITE =
  GBL_DISK_REM_FS_WRITE +
  GBL_DISK_REM_VM_WRITE +
  GBL_DISK_REM_SYSTEM_WRITE +
  GBL_DISK_REM_RAW_WRITE
```

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE_BYTE

The number of physical write KBs (or MBs if specified) during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE_BYTE_CUM

The number of physical write KBs (or MBs if specified) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE_CUM

The total number of physical writes over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE_PCT

The percentage of physical writes of total remote physical IO during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE_PCT_CUM

The percentage of physical writes of total remote physical IO over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE_RATE

The number of remote physical writes per second during the interval. All types of remote physical writes, including VM and raw, are counted.

This is calculated as

```
GBL_DISK_REM_PHYS_WRITE_RATE =
  GBL_DISK_REM_FS_WRITE_RATE +
  GBL_DISK_REM_VM_WRITE_RATE +
  GBL_DISK_REM_SYSTEM_WRITE_RATE +
  GBL_DISK_REM_RAW_WRITE_RATE
```

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_PHYS_WRITE_RATE_CUM

The number of physical writes per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

GBL_DISK_REM_RAW_BYTE

The number of remote KBs (or MBs if specified) transferred to or from a raw disk during the interval. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

GBL_DISK_REM_RAW_BYTE_CUM

The number of remote KBs (or MBs if specified) transferred to or from a raw disk over the cumulative collection time. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_REM_RAW_IO

The number of remote raw IOs during the interval. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

GBL_DISK_REM_RAW_IO_CUM

The total number of remote raw IOs over the cumulative collection time. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_REM_RAW_IO_PCT

The percentage of remote raw IOs to total remote physical disk IOs made during the interval. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

GBL_DISK_REM_RAW_IO_PCT_CUM

The percentage of remote raw IOs to total remote physical disk IOs made over the cumulative collection time. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_REM_RAW_IO_RATE

The total number of remote raw IOs per second during the interval. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

GBL_DISK_REM_RAW_IO_RATE_CUM

The average number of remote raw IOs over the cumulative collection time. On HP-UX, remote raw disk IO typically occurs when a client accesses a server disk in raw mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_REM_SYSTEM_BYTE

The number of remote KBs (or MBs if specified) transferred by the kernel from or to the remote machine for file system management access or updates during the interval.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_SYSTEM_BYTE_CUM

The number of remote KBs (or MBs if specified) transferred by the kernel to or from the remote machine for file system management access or updates over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_SYSTEM_IO

The number of remote physical IOs generated by the kernel for file system management (inode accesses or updates) during the interval.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_SYSTEM_IO_CUM

The number of remote physical reads and writes generated by the kernel for file system management (inode accesses or updates) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_SYSTEM_IO_PCT

The percentage of remote physical IOs generated by the kernel for file system management (inode accesses or updates) to the total number of remote physical IOs during the interval.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_SYSTEM_IO_PCT_CUM

The percentage of remote physical IOs generated by the kernel for file system management (inode updates) to the total number of remote physical disk IOs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_SYSTEM_IO_RATE

The number of remote physical IOs per second generated by the kernel for file system management (inode accesses or updates) during the interval.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_SYSTEM_IO_RATE_CUM

The number of remote physical reads and writes per second generated by the kernel over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are IOs for file system management (inode access or updates) and do not include IOs to user data.

On HP-UX, remote file system IO typically occurs during client file system access of a network file system mounted on the server. A remote file system IO does not necessarily imply that a physical IO occurs on the remote (server) system.

GBL_DISK_REM_VM_BYTE

The number of remote virtual memory KBs (or MBs if specified) transferred to or from the remote machine during the interval.

User file data transfers are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REM_VM_BYTE_CUM

The number of remote virtual memory KBs (or MBs if specified) transferred to or from the remote machine over the cumulative collection time. These are bytes transferred due to paging or swapping.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

User file data transfers are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REM_VM_IO

The total number of remote virtual memory IOs made during the interval. These are physical IOs related to paging or swapping.

IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REM_VM_IO_CUM

The total number of remote virtual memory IOs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REM_VM_IO_PCT

The percentage of remote virtual memory IO requests of total remote physical IOs during the interval.

IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REM_VM_IO_PCT_CUM

The percentage of remote virtual memory IOs of the total number of remote physical IOs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REM_VM_IO_RATE

The number of remote virtual memory IOs per second made during the interval. These are physical IOs related to paging, swapping, or memory mapped file allocations.

IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REM_VM_IO_RATE_CUM

The number of remote virtual memory IOs per second made over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, remote VM IO is typically seen on a client system that is paging in text from or paging out data pages to a server system. Paging in from the server system can occur when the client is loading a program which requires the text pages to be fetched from the server. Paging out occurs when client system data pages are swapped out to a remote swap device on the server system.

GBL_DISK_REQUEST_QUEUE

The total length of all of the disk queues at the end of the interval.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the “by-disk” data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_SUBSYSTEM_QUEUE

The average number of processes or kernel threads blocked on the disk subsystem (in a “queue” waiting for their file system disk IO to complete) during the interval. This is the sum of processes or kernel threads in the DISK, INODE, CACHE and CDFS wait states. Processes or kernel threads doing raw IO to a disk are not included in this measurement. As this number rises, it is an indication of a disk bottleneck.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on (DISK + INODE + CACHE + CDFS) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_DISK_SUBSYSTEM_WAIT_PCT

The percentage of time processes or kernel threads were blocked on the disk subsystem (waiting for their file system IOs to complete) during the interval. This is the sum of processes or kernel threads in the DISK, INODE, CACHE and CDFS wait states.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on (DISK + INODE + CACHE + CDFS) divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_DISK_SYSTEM_BYTE

The number of KBs (or MBs if specified) transferred by the kernel from or to the disk for file system management access or updates during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

GBL_DISK_SYSTEM_BYTE_CUM

The number of KBs (or MBs if specified) transferred by the kernel to or from disk for file system management access or updates over the cumulative collection time.

Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

GBL_DISK_SYSTEM_IO

The number of physical disk IOs generated by the kernel for file system management (inode accesses or updates) during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_SYSTEM_IO_CUM

The number of physical disk IOs generated by the kernel for file system management (inode accesses or updates) over the cumulative collection time.

Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_SYSTEM_IO_PCT

The percentage of physical disk IOs generated by the kernel for file system management (inode accesses or updates) to the total number of physical disk IOs during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_SYSTEM_IO_PCT_CUM

The percentage of physical IOs generated by the kernel for file system management (inode updates) to the total number of physical disk IOs over the cumulative collection time.

Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_SYSTEM_IO_RATE

The number of physical disk IOs per second generated by the kernel for file system management (inode accesses or updates) during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_SYSTEM_IO_RATE_CUM

The number of physical disk IOs per second generated by the kernel for file system management (inode accesses or updates) over the cumulative collection time. This rate does not include IOs to user data.

Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_SYSTEM_READ

Number of physical disk reads generated by the kernel for file system management (inode accesses) during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_SYSTEM_READ_RATE

Number of physical disk reads per second generated by the kernel for file system management (inode accesses) during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_SYSTEM_WRITE

Number of physical disk writes generated by the kernel for file system management (inode updates) during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_SYSTEM_WRITE_RATE

Number of physical disk writes per second generated by the kernel for file system management (inode updates) during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_TIME_PEAK

The time, in seconds, during the interval that the busiest disk was performing IO transfers. This is for the busiest disk only, not all disk devices. This counter is based on an end-to-end measurement for each IO transfer updated at queue entry and exit points.

Only local disks are counted in this measurement. NFS devices are excluded.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_UTIL

On HP-UX, this is the average percentage of time during the interval that all disks had IO in progress from the point of view of the Operating System. This is the average utilization for all disks.

On all other Unix systems, this is the average percentage of disk in use time of the total interval (that is, the average utilization).

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_UTIL_PEAK

The utilization of the busiest disk during the interval.

On HP-UX, this is the percentage of time during the interval that the busiest disk device had IO in progress from the point of view of the Operating System.

On all other systems, this is the percentage of time during the interval that the busiest disk was performing IO transfers.

It is not an average utilization over all the disk devices. Only local disks are counted in this measurement. NFS devices are excluded.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be “na” on the affected kernels. The “sar -d” command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

A peak disk utilization of more than 50 percent often indicates a disk IO subsystem bottleneck situation. A bottleneck may not be in the physical disk drive itself, but elsewhere in the IO path.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_UTIL_PEAK_CUM

The average utilization of the busiest disk in each interval over the cumulative collection time. Utilization is the percentage of time in use versus the time in the measurement interval. For each interval a different disk may be the busiest. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_UTIL_PEAK_HIGH

The highest utilization of any disk during any interval over the cumulative collection time. Utilization is the percentage of time in use versus the time in the measurement interval. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_DISK_UTIL_PEAK_OTHERS

The non-VM IO percent of the total utilization percent of the busiest disk during the interval. Utilization is the percentage of time in use versus the time in the measurement interval. Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_UTIL_PEAK_VM

The VM IO percent of the total utilization percent of the busiest disk during the interval. Utilization is the percentage of time in use versus the time in the measurement interval. Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_VM_BYTE

The number of virtual memory KBs (or MBs if specified) transferred to or from the disk during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the user file data transfers are not included in this metric unless they were done via the mmap(2) system call.

GBL_DISK_VM_BYTE_CUM

The number of virtual memory KBs (or MBs if specified) transferred to or from the disk over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the user file data transfers are not included in this metric unless they were done via the mmap(2) system call.

GBL_DISK_VM_IO

The total number of virtual memory IOs made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On SUN, when a file is accessed, it is memory mapped by the operating system. Accesses generate virtual memory IOs. Reading a file generates block IOs as the file's inode information is cached. File writes are a combination of posting to memory mapped allocations (VM IOs) and posting updated inode information to disk (block IOs).

On SUN, this metric is calculated by subtracting raw and block IOs from physical IOs. Tape drive accesses are included in the raw IOs, but not in the physical IOs. Therefore, when tape drive accesses are occurring on a system, all virtual memory and raw IO is counted as raw IO. For example, you may see heavy raw IO occurring during system backup. Raw IOs for disks are counted in the physical IOs. To determine if the raw IO is tape access versus disk access, compare the global physical disk accesses to the total of raw, block, and VM IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_VM_IO_CUM

The total number of virtual memory IOs over the cumulative collection time.

Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On SUN, when a file is accessed, it is memory mapped by the operating system. Accesses generate virtual memory IOs. Reading a file generates block IOs as the file's inode information is cached. File writes are a combination of posting to memory mapped allocations (VM IOs) and posting updated inode information to disk (block IOs).

On SUN, this metric is calculated by subtracting raw and block IOs from physical IOs. Tape drive accesses are included in the raw IOs, but not in the physical IOs. Therefore, when tape drive accesses are occurring on a system, all virtual memory and raw IO is counted as raw IO. For example, you may see heavy raw IO occurring during system backup. Raw IOs for disks are counted in the physical IOs. To determine if the raw IO is tape access versus disk access, compare the global physical disk accesses to the total of raw, block, and VM IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_VM_IO_PCT

On HP-UX and AIX, this is the percentage of virtual memory IO requests of total physical disk IOs during the interval.

On the other Unix systems, this is the percentage of virtual memory IOs of the total number of physical IOs during the interval.

Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On SUN, when a file is accessed, it is memory mapped by the operating system. Accesses generate virtual memory IOs. Reading a file generates block IOs as the file's inode information is cached. File writes are a combination of posting to memory mapped allocations (VM IOs) and posting updated inode information to disk (block IOs).

On SUN, this metric is calculated by subtracting raw and block IOs from physical IOs. Tape drive accesses are included in the raw IOs, but not in the physical IOs. Therefore, when tape drive accesses are occurring on a system, all virtual memory and raw IO is counted as raw IO. For example, you may see heavy raw IO occurring during system backup. Raw IOs for disks are counted in the physical IOs. To determine if the raw IO is tape access versus disk access, compare the global physical disk accesses to the total of raw, block, and VM IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_VM_IO_PCT_CUM

The percentage of virtual memory IOs of the total number of physical IOs over the cumulative collection time.

Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On SUN, when a file is accessed, it is memory mapped by the operating system. Accesses generate virtual memory IOs. Reading a file generates block IOs as the file's inode information is cached. File writes are a combination of posting to memory mapped allocations (VM IOs) and posting updated inode information to disk (block IOs).

On SUN, this metric is calculated by subtracting raw and block IOs from physical IOs. Tape drive accesses are included in the raw IOs, but not in the physical IOs. Therefore, when tape drive accesses are occurring on a system, all virtual memory and raw IO is counted as raw IO. For example, you may see heavy raw IO occurring during system backup. Raw IOs for disks are counted in the physical IOs. To determine if the raw IO is tape access versus disk access, compare the global physical disk accesses to the total of raw, block, and VM IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_VM_IO_RATE

The number of virtual memory IOs per second made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On SUN, when a file is accessed, it is memory mapped by the operating system. Accesses generate virtual memory IOs. Reading a file generates block IOs as the file's inode information is cached. File writes are a combination of posting to memory mapped allocations (VM IOs) and posting updated inode information to disk (block IOs).

On SUN, this metric is calculated by subtracting raw and block IOs from physical IOs. Tape drive accesses are included in the raw IOs, but not in the physical IOs. Therefore, when tape drive accesses are occurring on a system, all virtual memory and raw IO is counted as raw IO. For example, you may see heavy raw IO occurring during system backup. Raw IOs for disks are counted in the physical IOs. To determine if the raw IO is tape access versus disk access, compare the global physical disk accesses to the total of raw, block, and VM IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_VM_IO_RATE_CUM

On HP-UX and AIX, this is the number of virtual memory IOs per second made over the cumulative collection time.

On the other Unix systems, the number of virtual memory IOs per second made over the cumulative collection time.

Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the IOs to user file data are not included in this metric unless they were done via the mmap(2) system call.

On SUN, when a file is accessed, it is memory mapped by the operating system. Accesses generate virtual memory IOs. Reading a file generates block IOs as the file's inode information is cached. File writes are a combination of posting to memory mapped allocations (VM IOs) and posting updated inode information to disk (block IOs).

On SUN, this metric is calculated by subtracting raw and block IOs from physical IOs. Tape drive accesses are included in the raw IOs, but not in the physical IOs. Therefore, when tape drive accesses are occurring on a system, all virtual memory and raw IO is counted as raw IO. For example, you may see heavy raw IO occurring during system backup. Raw IOs for disks are counted in the physical IOs. To determine if the raw IO is tape access versus disk access, compare the global physical disk accesses to the total of raw, block, and VM IOs. If the totals are the same, the raw IO activity is to a disk, floppy, or CD drive. Check physical IO data for each individual disk device to isolate a device. If the totals are different, there is raw IO activity to a non-disk device like a tape drive.

GBL_DISK_VM_READ

The number of virtual memory reads made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the reads to user file data are not included in this metric unless they were accessed via the mmap(2) system call.

On AIX System WPARs, this metric is NA.

GBL_DISK_VM_READ_CUM

The number of virtual memory reads made over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the reads to user file data are not included in this metric unless they were accessed via the mmap(2) system call.

GBL_DISK_VM_READ_RATE

The number of virtual memory reads per second made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the reads to user file data are not included in this metric unless they were accessed via the mmap(2) system call.

On AIX System WPARs, this metric is NA.

GBL_DISK_VM_READ_RATE_CUM

The average number of virtual memory reads per second made over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the reads to user file data are not included in this metric unless they were accessed via the mmap(2) system call.

GBL_DISK_VM_READ_RATE_HIGH

The highest number of virtual memory reads per second made during any interval over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the reads to user file data are not included in this metric unless they were accessed via the mmap(2) system call.

GBL_DISK_VM_WRITE

The number of virtual memory writes made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the writes to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On AIX System WPARs, this metric is NA.

GBL_DISK_VM_WRITE_CUM

The number of virtual memory writes made over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the writes to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

GBL_DISK_VM_WRITE_RATE

The number of virtual memory writes per second made during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, the writes to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On AIX System WPARs, this metric is NA.

GBL_DISK_VM_WRITE_RATE_CUM

The average number of virtual memory writes per second made over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the writes to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

GBL_DISK_VM_WRITE_RATE_HIGH

The highest number of virtual memory writes per second made during any interval over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the writes to user file data are not included in this metric unless they were done via the mmap(2) system call.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

GBL_DISK_WAIT_PCT

The percentage of time processes or kernel threads were blocked on DISK (waiting in a disk driver for their disk IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on DISK divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_DISK_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on DISK (waiting in a disk driver for their disk IO to complete) during the interval.

GBL_FS_SPACE_UTIL_PEAK

The percentage of occupied disk space to total disk space for the fullest file system found during the interval. Only locally mounted file systems are counted in this metric.

This metric can be used as an indicator that at least one file system on the system is running out of disk space.

On Unix systems, CDROM and PC file systems are also excluded. This metric can exceed 100 percent. This is because a portion of the file system space is reserved as a buffer and can only be used by root. If the root user has made the file system grow beyond the reserved buffer, the utilization will be greater than 100 percent. This is a dangerous situation since if the root user totally fills the file system, the system may crash.

On Windows, CDROM file systems are also excluded.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_GMTOFFSET

The difference, in minutes, between local time and GMT (Greenwich Mean Time).

GBL_GRAPHICS_QUEUE

The average number of processes or kernel threads blocked on graphics (waiting for their graphics operations to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on GRAPH (that is, graphics) divide by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_GRAPHICS_WAIT_PCT

The percentage of time processes or kernel threads were blocked on graphics (waiting for their graphics operations to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on GRAPH (that is, graphics) divide by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_GRAPHICS_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on graphics (waiting for their graphics operations to complete) during the interval.

GBL_IGNORE_MT

This boolean value indicates whether the CPU normalization is on or off. If the metric value is “true”, CPU related metrics in the global class will report values which are normalized against the number of active cores on the system.

If the metric value is “false”, CPU related metrics in the global class will report values which are normalized against the number of CPU threads on the system.

If CPU MultiThreading is turned off this configuration option is a no-op and the metric value will be “true”.

On Linux, this metric will only report “true” if this configuration is on and if the kernel provides enough information to determine whether MultiThreading is turned on.

On HPUX, this metric will report “na” if the processor doesn't support the feature.

GBL_INODE_QUEUE

The average number of processes or kernel threads blocked onINODE (waiting for an inode to be updated or to become available) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on INODE divided by the interval time.

Inodes are used to store information about files within the file system. Every file has at least two inodes associated with it (one for the directory and one for the file itself). The information stored in an inode includes the owners, timestamps, size, and an array of indices used to translate logical block numbers to physical sector numbers. There is a separate inode maintained for every view of a file, so if two processes have the same file open, they both use the same directory inode, but separate inodes for the file.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_INODE_WAIT_PCT

The percentage of time processes or kernel threads were blocked onINODE (waiting for an inode to be updated or to become available) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on INODE divided by the accumulated time that all processes or kernel threads were alive during the interval.

Inodes are used to store information about files within the file system. Every file has at least two inodes associated with it (one for the directory and one for the file itself). The information stored in an inode includes the owners, timestamps, size, and an array of indices used to translate logical block numbers to physical sector numbers. There is a separate inode maintained for every view of a file, so if two processes have the same file open, they both use the same directory inode, but separate inodes for the file.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_INODE_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked onINODE (waiting for an inode to be updated or to become available) during the interval.

GBL_INTERRUPT

The number of IO interrupts during the interval.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_INTERRUPT_RATE

The average number of IO interrupts per second during the interval.

On HPUX and SUN this value includes clock interrupts. To get non-clock device interrupts, subtract clock interrupts from the value.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_INTERRUPT_RATE_CUM

The average number of IO interrupts per second over the cumulative collection time.

On HPUX and SUN this value includes clock interrupts. To get non-clock device interrupts, subtract clock interrupts from the value.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_INTERRUPT_RATE_HIGH

The highest number of IO interrupts per second during any one interval over the cumulative collection time.

On HPUX and SUN this value includes clock interrupts. To get non-clock device interrupts, subtract clock interrupts from the value.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_INTERVAL

The amount of time in the interval.

This measured interval is slightly larger than the desired or configured interval if the collection program is delayed by a higher priority process and cannot sample the data immediately.

GBL_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_IPC_QUEUE

The average number of processes or kernel threads blocked onInterProcess Communication (IPC) (waiting for their interprocess communication calls to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on IPC divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_IPC_SUBSYSTEM_QUEUE

The average number of processes or kernel threads blocked on the InterProcess Communication (IPC) subsystems (waiting for their interprocess communication activity to complete) during the interval. This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on (IPC + MSG + SEM + PIPE + SOCKT + STRMS) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_IPC_SUBSYSTEM_WAIT_PCT

The percentage of time processes or kernel threads were blocked on the InterProcess Communication (IPC) subsystems (waiting for their interprocess communication activity to complete) during the interval. This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on (IPC + MSG + SEM + PIPE + SOCKT + STRMS) divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_IPC_WAIT_PCT

The percentage of time processes or kernel threads were blocked onInterProcess Communication (IPC) (waiting for their interprocess communication calls to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on IPC divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the

system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_IPC_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked onInterProcess Communication (IPC) (waiting for their interprocess communication calls to complete) during the interval.

GBL_JAVAARG

This boolean value indicates whether the java class overloading mechanism is enabled or not. This metric will be set when the javaarg flag in the parm file is set. The metric affected by this setting is PROC_PROC_ARGV1. This setting is useful to construct parm file java application definitions using the argv1= keyword.

GBL_JOBCTL_QUEUE

The average number of processes or kernel threads blocked on job control (having been stopped with the job control facilities) during the interval. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on job control divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_JOBCTL_WAIT_PCT

The percentage of time processes or kernel threads were blocked on job control (having been stopped with the job control facilities) during the interval. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on job control divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_JOBCTL_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on job control (having been stopped with the job control facilities) during the interval. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

GBL_LAN_QUEUE

The average number of processes or kernel threads blocked on LAN (waiting for their IO over the LAN to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on LAN divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_LAN_WAIT_PCT

The percentage of time processes or kernel threads were blocked on LAN (waiting for their IO over the LAN to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on LAN divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_LAN_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on LAN (waiting for their IO over the LAN to complete) during the interval.

GBL_LOADAVG

The 1 minute load average of the system obtained at the time of logging.

On windows this is the load average of the system over the interval. Load average on windows is the average number of threads that have been waiting in ready state during the interval. This is obtained by checking the number of threads in ready state every sub proc interval, accumulating them over the interval and averaging over the interval.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_LOADAVG15

The 15 minute load average of the system obtained at the time of logging.

GBL_LOADAVG5

The 5 minute load average of the system obtained at the time of logging.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_LOST_MI_TRACE_BUFFERS

The number of trace buffers lost by the measurement processing daemon.

On HP-UX systems, if this value is > 0, the measurement subsystem is not keeping up with the system events that generate traces.

For other Unix systems, if this value is > 0, the measurement subsystem is not keeping up with the ARM API calls that generate traces.

Note: The value reported for this metric will roll over to 0 once it crosses INTMAX.

GBL_LS_ROLE

Indicates whether Perf Agent is installed on Logical system or host or standalone system. This metric will be either “GUEST”, “HOST” or “STAND”.

GBL_LS_TYPE

The virtulization technology if applicable. The value of this metric is “HPVM” on HP-UX host, “LPAR” on AIX LPAR, “Sys WPAR” on system WPAR, “Zone” on Solaris Zones, “VMware” on recognized VMware ESX guest and VMware ESX Server console, “Hyper-V” on Hyper-V host, else “NoVM”.

In conjunction with GBL_LS_ROLE this metric could be used to identify the environment in which Perf Agent/Glance is running. For example, if GBL_LS_ROLE is “Guest” and GBL_LS_TYPE is “VMware” then PA/Glance is running on a VMware Guest.

GBL_LS_UUID

UUID of this logical system. This Id uniquely identifies this logical system in the virtualized enviroments. On a standalone system the value of this metrics is 'na'.

GBL_MACHINE

An ASCII string representing the Processor Architecture. And machine hardware model is represented by GBL_MACHINE_MODEL metric.

GBL_MACHINE_MODEL

The CPU model. This is similar to the information returned by the GBL_MACHINE metric and the uname command(except for Solaris 10 x86/x86_64). However, this metric returns more information on some processors.

On HP-UX, this is the same information returned by the model command.

GBL_MEMFS_BLK_CNT

The number of system memory blocks used by Memory based FileSystem (MemFS).

GBL_MEMFS_SWP_CNT

The number of system memory blocks swapped by Memory based FileSystem (MemFS).

GBL_MEM_ACTIVE_VIRT

The total virtual memory (in MBs unless otherwise specified) allocated for processes that are currently on the run queue or processes that have executed recently. This is the sum of the virtual memory sizes of the data and stack regions for these processes.

On HP-UX, this is the sum of the virtual memory of all processes which have had a thread run in the last 20 seconds.

On AIX System WPARs, this metric is NA.

GBL_MEM_ACTIVE_VIRT_UTIL

The percentage of total virtual memory active at the end of the interval.

Active virtual memory is the virtual memory associated with processes that are currently on the run queue or processes that have executed recently. This is the sum of the virtual memory sizes of the data and stack regions for these processes.

On HP-UX, this is the sum of the virtual memory of all processes which have had a thread run in the last 20 seconds.

GBL_MEM_AVAIL

The amount of physical available memory in the system (in MBs unless otherwise specified).

On Windows, memory resident operating system code and data is not included as available memory.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_CACHE

The amount of physical memory (in MBs unless otherwise specified) used by the buffer cache during the interval.

On HP-UX 11i v2 and below, the buffer cache is a memory pool used by the system to stage disk IO data for the driver.

On HP-UX 11i v3 and above this metric value represents the usage of the file systembuffer cache which is still being used for file system metadata.

On SUN, this value is obtained by multiplying the system page size times the number of buffer headers (nbuf). For example, on a SPARCstation 10 the buffer size is usually (200 (page size buffers) * 4096 (bytes/page) = 800 KB). If ZFS is configured, this includes ZFS ARC cache usage during the interval.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On AIX, this value should be minimal since most disk IOs are done through memory mapped files.

GBL_MEM_CACHE_HIT

On HP-UX, the number of buffer cache reads resolved from the buffer cache (rather than going to disk) during the interval. Buffer cache reads can occur as a result of a logical read (for example, file read system call), a read generated by a client, a read-ahead on behalf of a logical read or a system procedure.

On HP-UX, this metric is obtained by measuring the number of buffered read calls that were satisfied by the data that was in the file system buffer cache. Reads that are not in the buffer cache result in disk IO. raw IO and virtual memory IO, are not counted in this metric.

On SUN, the number of physical reads resolved from memory (rather than going to disk) during the interval. This includes inode, indirect block and cylinder group related disk reads, plus file reads from files memory mapped by the virtual memory IO system.

On AIX, the number of disk reads that were satisfied in the file systembuffer cache (rather than going to disk) during the interval.

On AIX, the traditional file system buffer cache is not normally used, since files are implicitly memory mapped and the access is through the virtual memory system rather than the buffer cache. However, if a file is read as a block device (e.g /dev/hdisk1), the file system buffer cache is used, making this metric meaningful in that situation. If no IO through the buffer cache occurs during the interval, this metric is 0.

GBL_MEM_CACHE_HIT_CUM

On HP-UX, the number of buffer cache reads resolved from the buffer cache (rather than going to disk) over the cumulative collection time. Buffer cache reads can occur as a result of a logical read (for example, file read system call), a read generated by a client, a read-ahead on behalf of a logical read or a system procedure.

On HP-UX, this metric is obtained by measuring the number of buffered read calls that were satisfied by the data that was in the file system buffer cache. Reads that are not in the buffer cache result in disk IO. raw IO and virtual memory IO, are not counted in this metric.

On SUN, the number of physical reads resolved from memory (rather than going to disk) over the cumulative collection time. This includes inode, indirect block and cylinder group related disk reads, plus file reads from files memory mapped by the virtual memory IO system.

On AIX, the number of disk reads that were satisfied in the file systembuffer cache (rather than going to disk) over the cumulative collection time.

On AIX, the traditional file system buffer cache is not normally used, since files are implicitly memory mapped and the access is through the virtual memory system rather than the buffer cache. However, if a file is read as a block device (e.g /dev/hdisk1), the file system buffer cache is used, making this metric meaningful in that situation. If no IO through the buffer cache occurs during the interval, this metric is 0.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_CACHE_HIT_PCT

On HP-UX, the percentage of buffer cache reads resolved from the buffer cache (rather than going to disk) during the interval. Buffer cache reads can occur as a result of a logical read (for example, file read system call), a read generated by a client, a read-ahead on behalf of a logical read or a system procedure.

On HP-UX, this metric is obtained by measuring the number of buffered read calls that were satisfied by the data that was in the file system buffer cache. Reads to filesystem file buffers that are not in the buffer cache result in disk IO. Reads to raw IO and virtual memory IO (including memory mapped files), do not go through the filesystem buffer cache, and so are not relevant to this metric.

On HP-UX, a low cache hit rate may indicate low efficiency of the buffer cache, either because applications have poor data locality or because the buffer cache is too small. Overly large buffer cache sizes can lead to a memory bottleneck. The buffer cache should be sized small enough so that pageouts do not occur even when the system is busy. However, in the case of VxFS, all memory-mapped IOs show up as page ins/page outs and are not a result of memory pressure.

On AIX, the percentage of disk reads that were satisfied in the file systembuffer cache (rather than going to disk) during the interval.

On AIX, the traditional file system buffer cache is not normally used, since files are implicitly memory mapped and the access is through the virtual memory system rather than the buffer cache. However, if a file is read as a block device (e.g /dev/hdisk1), the file system buffer cache is used, making this metric meaningful in that situation. If no IO through the buffer cache occurs during the interval, this metric is 0.

On the remaining Unix systems, this is the percentage of logical reads satisfied in memory (rather than going to disk) during the interval. This includes inode, indirect block and cylinder group related disk reads, plus file reads from files memory mapped by the virtual memory IO system.

On Windows, this is the percentage of buffered reads satisfied in the buffer cache (rather than going to disk) during the interval. This metric is obtained by measuring the number of buffered read calls that were satisfied by the data that was in the system buffer cache. Reads that are not in the buffer cache result in disk IO. Unbuffered IO and virtual memory IO (including memory mapped files), are not counted in this metric.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_MEM_CACHE_HIT_PCT_CUM

On HP-UX, this is the average percentage of buffer cache reads resolved from the buffer cache (rather than going to disk) over the cumulative collection time. Buffer cache reads can occur as a result of a logical read (for example, file read system call), a read generated by a client, a read-ahead on behalf of a logical read or a system procedure.

On SUN, this is the percentage of physical reads that were satisfied in memory (rather than going to disk) over the cumulative collection time. This includes inode, indirect block and cylinder group related disk reads, plus file reads from files memory mapped by the virtual memory IO system.

On AIX, this is the percentage of physical reads satisfied in the file systembuffer cache (rather than going to disk) over the cumulative collection time.

On AIX, the traditional file system buffer cache is not normally used, since files are implicitly memory mapped and the access is through the virtual memory system rather than the buffer cache. However, if a file is read as a block device (e.g /dev/hdisk1), the file system buffer cache is used, making this metric meaningful in that situation. If no IO through the buffer cache occurs during the interval, this metric is 0.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_CACHE_HIT_PCT_HIGH

On HP-UX, this is the highest interval percentage of buffer cache reads resolved from the buffer cache (rather than going to disk) over the cumulative collection time. Buffer cache reads can occur as a result of a logical read (for example, file read system call), a read generated by a client, a read-ahead on behalf of a logical read or a system procedure.

On SUN, this is the highest interval percentage of physical reads satisfied in memory (rather than going to disk) over the cumulative collection time. This includes inode, indirect block and cylinder group related disk reads, plus file reads from files memory mapped by the virtual memory IO system.

On AIX, this is the highest interval percentage of physical reads satisfied in the file systembuffer cache (rather than going to disk) over the cumulative collection time.

On AIX, the traditional file system buffer cache is not normally used, since files are implicitly memory mapped and the access is through the virtual memory system rather than the buffer cache. However, if a file is read as a block device (e.g /dev/hdisk1), the file system buffer cache is used, making this metric meaningful in that situation. If no IO through the buffer cache occurs during the interval, this metric is 0.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_CACHE_UTIL

The percentage of physical memory used by the buffer cache during the interval.

On HP-UX 11i v2 and below, the buffer cache is a memory pool used by the system to stage disk IO data for the driver.

On HP-UX 11i v3 and above this metric value represents the usage of the file systembuffer cache which is still being used for file system metadata.

On SUN, this percentage is based on calculating the buffer cache size by multiplying the system page size times the number of buffer headers (nbuf). For example, on a SPARCstation 10 the buffer size is usually (200 (page size buffers) * 4096 (bytes/page) = 800 KB). If ZFS is configured, this includes ZFS ARC cache utilization during the interval.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On AIX, this value should be minimal since most disk IOs are done through memory mapped files. On Windows the value reports 'copy read hit %' and 'Pin read hit %'.

GBL_MEM_CACHE_WRITE_HIT

The number of write cache hits - logical writes that did not result in physical IOs during the interval.

A cache write hit occurs when a logical write request is issued to a disk file block that is already mapped in a buffer that is in a delayed write state. This metric gives an indication of how many physical IOs are eliminated as a result of buffering logical write requests. Physical IOs are eliminated in environments where asynchronous writes are done (see the O_SYNC flag in open(2)) to the same file blocks before being explicitly written to the disk or flushed to disk by the syncher process. Environments that attempt to minimize the chance of file system data loss by issuing synchronous writes or by using shorter syncer intervals will see fewer cache write hits.

During a short interval, the number of physical writes can exceed the number of logical write requests. This would yield a negative number of “write hits”. If this occurs in an interval, “na” will be returned.

GBL_MEM_CACHE_WRITE_HIT_CUM

The number of write cache hits - logical writes that did not result in physical IOs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

A cache write hit occurs when a logical write request is issued to a disk file block that is already mapped in a buffer that is in a delayed write state. This metric gives an indication of how many physical IOs are eliminated as a result of buffering logical write requests. Physical IOs are eliminated in environments where asynchronous writes are done (see the O_SYNC flag in open(2)) to the same file blocks before being explicitly written to the disk or flushed to disk by the syncher process. Environments that attempt to minimize the chance of file system data loss by issuing synchronous writes or by using shorter syncer intervals will see fewer cache write hits.

GBL_MEM_CACHE_WRITE_HIT_PCT

The percentage of logical disk writes that did not result in physical disk IOs during the interval.

A cache write hit occurs when a logical write request is issued to a disk file block that is already mapped in a buffer that is in a delayed write state. This metric gives an indication of how many physical IOs are eliminated as a result of buffering logical write requests. Physical IOs are eliminated in environments where asynchronous writes are done (see the O_SYNC flag in open(2)) to the same file blocks before being explicitly written to the disk or flushed to disk by the syncher process. Environments that attempt to minimize the chance of file system data loss by issuing synchronous writes or by using shorter syncer intervals will see fewer cache write hits.

During a short interval, the number of physical writes can exceed the number of logical write requests. This would yield a negative number of “write hits”. If this occurs in an interval, “na” will be returned.

GBL_MEM_CACHE_WRITE_HIT_PCT_CUM

The percentage of logical disk writes that did not result in physical disk IOs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

A cache write hit occurs when a logical write request is issued to a disk file block that is already mapped in a buffer that is in a delayed write state. This metric gives an indication of how many physical IOs are eliminated as a result of buffering logical write requests. Physical IOs are eliminated in environments where asynchronous writes are done (see the O_SYNC flag in open(2)) to the same file blocks before being explicitly written to the disk or flushed to disk by the syncher process. Environments that attempt to minimize the chance of file system data loss by issuing synchronous writes or by using shorter syncer intervals will see fewer cache write hits.

GBL_MEM_DNLC_HIT

The number of times a pathname component was found in the directory name lookup cache (rather than requiring a disk read to find a file) during the interval.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
ncsize = MAX(((nproc+16+maxusers)+
            32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_HIT_CUM

The number of times a pathname component was found in the directory name lookup cache (rather than requiring a disk read to find a file) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
  ncsize = MAX(((nproc+16+maxusers)+
             32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_HIT_PCT

The percentage of time a pathname component was found in the directory name lookup cache (rather than requiring a disk read to find a file) during the interval.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable "ncsize" and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the "ncsize". The formula is:

```
  ncsize = MAX(((nproc+16+maxusers)+
             32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. "Enters", or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

On Solaris non-global zones, this metric shows data from the global zone.

GBL_MEM_DNLC_HIT_PCT_CUM

The percentage of time a pathname component was found in the directory name lookup cache (rather than requiring a disk read to find a file) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
ncsize = MAX(((nproc+16+maxusers)+
          32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_HIT_PCT_HIGH

The highest percentage of time during any one interval that a pathname component was found in the directory name lookup cache (rather than requiring a disk read to find a file) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode

cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
  ncsize = MAX(((nproc+16+maxusers)+
             32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_LONGS

The number of times a pathname component was too long to be found in the directory name lookup cache during the interval.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
  ncsize = MAX(((nproc+16+maxusers)+
             32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_LONGS_CUM

The number of times a pathname component was too long to be found in the directory name lookup cache over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode

cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
ncsize = MAX(((nproc+16+maxusers)+
          32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_LONGS_PCT

The percentage of time a pathname component was too long to be found in the directory name lookup cache during the interval.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
  ncsize = MAX(((nproc+16+maxusers)+
              32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_LONGS_PCT_CUM

The percentage of time a pathname component was too long to be found in the directory name lookup cache over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode

cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
 ncsize = MAX(((nproc+16+maxusers)+
           32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_DNLC_LONGS_PCT_HIGH

The highest percentage of time during any one interval that a pathname component was too long to be found in the directory name lookup cache over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
ncsize = MAX(((nproc+16+maxusers)+
           32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

On SUN, long file names (greater than 30 characters) are not cached and are a type of cache miss. “Enters”, or cache data updates, are not included in this data. The DNLC size is: (maxusers * 17) + 90

GBL_MEM_FILE_PAGE_CACHE

The amount of physical memory (in MBs unless otherwise specified) used by the file cache during the interval. File cache is a memory pool used by the system to stage disk IO data for the driver.

This metric is supported on HP-UX 11iv3 and above. The filecache_min and filecache_max tunables control the filecache memory usage on the system. The filecache_min tunable specifies the amount of physical memory that is guaranteed to be available for filecache on the system. The filecache memory usage can grow beyond filecache_min, up to the limit set by the filecache_max tunable. The Virtual Memory(VM) subsystem always pre reserves 'filecache_min' tunable value worth of pages on the system for filecache, even in the case of filecache under utilization (actual filecache utilization < filecache_min value). This preserved memory by the VM is not available for the user. In this scenario, this metric will show the 'filecache_min' as the filecache value, rather than showing the actual filecache utilization.

On Linux, this metric is equal to 'cached' value of 'free -m' command output.

GBL_MEM_FILE_PAGE_CACHE_UTIL

The percentage of physical_memory used by the file cache during the interval. File cache is a memory pool used by the system to stage disk IO data for the driver.

This metric is supported on HP-UX 11iv3 and above. The filecache_min and filecache_max tunables control the filecache memory usage on the system. The filecache_min tunable specifies the amount of physical memory that is guaranteed to be available for filecache on the system. The filecache memory usage can grow beyond filecache_min, up to the limit set by the filecache_max tunable. The Virtual Memory(VM) subsystem always pre reserves 'filecache_min' tunable value worth of pages on the system for filecache, even in the case of filecache under utilization (actual filecache utilization < filecache_min value). This preserved memory by the VM is not available for the user. In this scenario, this metric will show the 'filecache_min' as the filecache value, rather than showing the actual filecache utilization.

On Linux, this metric is derived from 'cached' value of 'free -m' command output.

GBL_MEM_FREE

The amount of memory not allocated (in MBs unless otherwise specified). As this value drops, the likelihood increases that swapping or paging out to disk may occur to satisfy new memory requests.

On SUN, low values for this metric may not indicate a true memory shortage. This metric can be influenced by the VMM (Virtual Memory Management) system. On uncapped solaris zones, the metric indicates the amount of memory that is available across the whole system that is not consumed by the global zone and other non-global zones. In case of capped solaris zones, the metric indicates the amount of memory that is not consumed by this zone against the memory cap set.

On Linux, this metric is sum of 'free' and 'cached' memory.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

Locality Domain metrics are available on HP-UX 11iv2 and above. GBL_MEM_FREE and LDOM_MEM_FREE, as well as the memory utilization metrics derived from them, may not always fully match. GBL_MEM_FREE represents free memory in the kernel's reservation layer while LDOM_MEM_FREE shows actual free pages. If memory has been reserved but not actually consumed from the Locality Domains, the two values won't match. Because GBL_MEM_FREE includes pre-reserved memory, the GBL_MEM_* metrics are a better indicator of actual memory consumption in most situations.

GBL_MEM_FREE_UTIL

The percentage of physical memory that was free at the end of the interval.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEIN

The total number of page ins from the disk during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX, this is the same as the "page ins" value from the "vmstat -s" command. On AIX, this is the same as the "paging space page ins" value. Remember that "vmstat -s" reports cumulative counts.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEIN_BYTE

The number of KBs (or MBs if specified) of page ins during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_CUM

The number of KBs (or MBs if specified) of page ins over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_RATE

The number of KBs per second of page ins during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_RATE_CUM

The average number of KBs per second of page ins over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_RATE_HIGH

The highest number of KBs per second of page ins during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_CUM

The total number of page ins from the disk over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_RATE

The total number of page ins per second from the disk during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX and AIX, this is the same as the “pi” value from the vmstat command.

On Solaris, this is the same as the sum of the “epi” and “api” values from the “vmstat -p” command, divided by the page size in KB.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEIN_RATE_CUM

The average number of page ins per second over the cumulative collection time. This includes pages paged in from paging space and, except for AIX, from the file system.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_RATE_HIGH

The highest number of page ins per second from disk during any interval over the cumulative collection time.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT

The total number of page outs to the disk during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX, this is the same as the “page outs” value from the “vmstat -s” command. On HP-UX 11iv3 and above this includes filecache page outs also. On AIX, this is the same as the “paging space page outs” value. Remember that “vmstat -s” reports cumulative counts.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_BYTE

The number of KBs (or MBs if specified) of page outs during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_BYTE_CUM

The number of KBs (or MBs if specified) of page outs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_BYTE_RATE

The number of KBs (or MBs if specified) per second of page outs during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_BYTE_RATE_CUM

The average number of KBs per second of page outs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_BYTE_RATE_HIGH

The highest number of KBs per second of page outs during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_CUM

The total number of page outs to the disk over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_RATE

The total number of page outs to the disk per second during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX and AIX, this is the same as the “po” value from the vmstat command.

On Solaris, this is the same as the sum of the “epo” and “apo” values from the “vmstat -p” command, divided by the page size in KB.

On Windows, this counter also includes paging traffic on behalf of the system cache to access file data for applications and so may be high when there is no memory pressure.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_RATE_CUM

The average number of page outs to the disk per second over the cumulative collection time. This includes pages paged out to paging space and, except for AIX, to the file system.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_RATE_HIGH

The highest number of page outs per second to disk during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGE_FAULT

The number of page faults that occurred during the interval.

On Linux this metric is available only on 2.6 and above kernel versions.

GBL_MEM_PAGE_FAULT_CUM

The number of page faults that occurred over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_PAGE_FAULT_RATE

The number of page faults per second during the interval.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGE_FAULT_RATE_CUM

The average number of page faults per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_PAGE_FAULT_RATE_HIGH

The highest page fault per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_PAGE_REQUEST

The number of page requests to or from the disk during the interval.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

On HP-UX, this is the same as the sun of the “page ins” and “page outs” values from the “vmstat -s” command. On AIX, this is the same as the sum of the “paging space page ins” and “paging space page outs” values. Remember that “vmstat -s” reports cumulative counts.

On Windows, this counter also includes paging traffic on behalf of the system cache to access file data for applications and so may be high when there is no memory pressure.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGE_REQUEST_CUM

The total number of page requests to or from the disk over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

On Windows, this counter also includes paging traffic on behalf of the system cache to access file data for applications and so may be high when there is no memory pressure.

GBL_MEM_PAGE_REQUEST_RATE

The number of page requests to or from the disk per second during the interval.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

On HP-UX and AIX, this is the same as the sum of the “pi” and “po” values from the vmstat command.

On Solaris, this is the same as the sum of the “epi”, “epo”, “api”, and “apo” values from the “vmstat -p” command, divided by the page size in KB.

Higher than normal rates can indicate either a memory or a disk bottleneck. Compare GBL_DISK_UTIL_PEAK and GBL_MEM_UTIL to determine which resource is more constrained. High rates may also indicate memory thrashing caused by a particular application or set of applications. Look for processes with high major fault rates to identify the culprits.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGE_REQUEST_RATE_CUM

The average number of page requests to or from the disk per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

GBL_MEM_PAGE_REQUEST_RATE_HIGH

The highest number of page requests per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

GBL_MEM_PAGE_SIZE_MAX

The maximum page size allowed for a memory region on the system.

GBL_MEM_PG_SCAN

The number of pages scanned by the pageout daemon (or by the Clock Hand on AIX) during the interval. The clock hand algorithm is used to control page aging on the system.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PG_SCAN_CUM

The number of pages scanned by the pageout daemon (or by the Clock Hand on AIX) over the cumulative collection time. The clock hand algorithm is used to control page aging on the system.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_PG_SCAN_RATE

The number of pages scanned per second by the pageout daemon (or by the Clock Hand on AIX, “vmstat -s” pages examined by clock) during the interval. The clock hand algorithm is used to control page aging on the system.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PG_SCAN_RATE_CUM

The average number of pages scanned per second by the pageout daemon (or by the Clock Hand on AIX) over the cumulative collection time. The clock hand algorithm is used to control page aging on the system.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_PG_SCAN_RATE_HIGH

The highest number of pages scanned per second by the pageout daemon (or by the Clock Hand on AIX) during any interval over the cumulative collection time. The clock hand algorithm is used to control page aging on the system.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_PHYS

The amount of physical memory in the system (in MBs unless otherwise specified).

On HP-UX, banks with bad memory are not counted. Note that on some machines, the Processor Dependent Code (PDC) code uses the upper 1MB of memory and thus reports less than the actual physical memory of the system. Thus, on a system with 256MB of physical memory, this metric and dmesg(1M) might only report 267,386,880 bytes (255MB). This is all the physical memory that software on the machine can access.

On Windows, this is the total memory available, which may be slightly less than the total amount of physical memory present in the system. This value is also reported in the Control Panel's About Windows NT help topic.

On Linux, this is the amount of memory given by dmesg(1M). If the value is not available in kernel ring buffer, then the sum of system memory and available memory will be reported as physical memory.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_QUEUE

The average number of processes or kernel threads blocked on memory (waiting for virtual memory disk accesses to complete) during the interval. This typically happens when processes or kernel threads are allocating a large amount of memory. It can also happen when processes or kernel threads access memory that has been paged out to disk (swap) because of overall memory pressure on the system. Note that large programs can block on VM disk access when they are initializing, bringing their text and data pages into memory. When this metric rises, it can be an indication of a memory bottleneck, especially if overall system memory utilization (GBL_MEM_UTIL) is near 100% and there is also swapout or page out activity.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on memory divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_MEM_SWAP

The total number of swap ins and swap outs (or deactivations and reactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN

The number of swap ins (or reactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, this is the same as the “swap ins” value from the “vmstat -s” command. Remember that “vmstat -s” reports cumulative counts.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_BYTE

The number of KBs transferred in from disk due to swap ins (or reactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPIN_BYTE_CUM

The number of KBs transferred in from disk due to swap ins (or reactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_BYTE_RATE

The number of KBs per second transferred from disk due to swap ins (or reactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPIN_BYTE_RATE_CUM

The number of KBs per second transferred from disk due to swap ins (or reactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_BYTE_RATE_HIGH

The highest number of KBs per second transferred from disk due to swap ins (or reactivations on HP-UX) during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in

bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_CUM

The number of swap ins (or reactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_RATE

The number of swap ins (or reactivations on HP-UX) per second during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPIN_RATE_CUM

The average number of swap ins (or reactivations on HP-UX) per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in

bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_RATE_HIGH

The highest number of swap ins (or reactivations on HP-UX) per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT

The number of swap outs (or deactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, this is the same as the “swap outs” values from the “vmstat -s” command. Remember that “vmstat -s” reports cumulative counts.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_BYTE

The number of KBs (or MBs if specified) transferred out to disk due to swap outs (or deactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPOUT_BYTE_CUM

The number of KBs (or MBs if specified) transferred out to disk due to swap outs (or deactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_BYTE_RATE

The number of KBs (or MBs if specified) per second transferred out to disk due to swap outs (or deactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in

bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPOUT_BYTE_RATE_CUM

The average number of KBs (or MBs if specified) per second transferred out to disk due to swap outs (or deactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_BYTE_RATE_HIGH

The highest number of KBs (or MBs if specified) per second transferred out to disk due to swap outs (or deactivations on HP-UX) during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_CUM

The number of swap outs (or deactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_RATE

The number of swap outs (or deactivations on HP-UX) per second during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPOUT_RATE_CUM

The number of swap outs (or deactivations on HP-UX) per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_RATE_HIGH

The highest number of swap outs (or deactivations on HP-UX) per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAP_1_MIN_RATE

The number of swap ins and swap outs (or deactivations/reactivations on HP-UX) per minute during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAP_CUM

The total number of swap ins and swap outs (or deactivations and reactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAP_RATE

The total number of swap ins and swap outs (or deactivations and reactivations on HP-UX) per second during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAP_RATE_CUM

The average number of swap ins and swap outs (or deactivations and reactivations on HP-UX) per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAP_RATE_HIGH

The highest number of swap ins and swap outs (or deactivations and reactivations on HP-UX) per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SYS

The amount of physical memory (in MBs unless otherwise specified) used by the system (kernel) during the interval. System memory does not include the buffer cache. On HP-UX and Linux this does not include filecache also.

On HP-UX 11.0, this metric does not include some kinds of dynamically allocated kernel memory. This has always been reported in the GBL_MEM_USER* metrics.

On HP-UX 11.11 and beyond, this metric includes some kinds of dynamically allocated kernel memory.

On Solaris non-global zones, this metric shows value as 0.

GBL_MEM_SYS_AND_CACHE_UTIL

The percentage of physical memory used by the system (kernel) and the buffer cache at the end of the interval.

On HP-UX 11iv3, this includes file cache also.

On HP-UX 11.0, this metric does not include some kinds of dynamically allocated kernel memory. This has always been reported in the GBL_MEM_USER* metrics.

On HP-UX 11.11 and beyond, this metric includes some kinds of dynamically allocated kernel memory.

On Solaris non-global zones, this metric is N/A.

GBL_MEM_SYS_UTIL

The percentage of physical memory used by the system during the interval.

System memory does not include the buffer cache. On HP-UX and Linux this does not include filecache also.

On HP-UX 11.0, this metric does not include some kinds of dynamically allocated kernel memory. This has always been reported in the GBL_MEM_USER* metrics.

On HP-UX 11.11 and beyond, this metric includes some kinds of dynamically allocated kernel memory.

On Solaris non-global zones, this metric shows value as 0.

GBL_MEM_USER

The amount of physical memory (in MBs unless otherwise specified) allocated to user code and data at the end of the interval. User memory regions include code, heap, stack, and other data areas including shared memory. This does not include memory for buffer cache. On HP-UX and Linux this does not include filecache also.

On HP-UX 11.0, this metric includes some kinds of dynamically allocated kernel memory.

On HP-UX 11.11 and beyond, this metric does not include some kinds of dynamically allocated kernel memory. This is now reported in the GBL_MEM_SYS* metrics.

Large fluctuations in this metric can be caused by programs which allocate large amounts of memory and then either release the memory or terminate. A slow continual increase in this metric may indicate a program with a memory leak.

GBL_MEM_USER_UTIL

The percent of physical memory allocated to user code and data at the end of the interval. This metric shows the percent of memory owned by user memory regions such as user code, heap, stack and other data areas including shared memory. This does not include memory for buffer cache. On HP-UX and Linux this does not include filecache also. On HP-UX 11.0, this metric includes some kinds of dynamically allocated kernel memory.

On HP-UX 11.11 and beyond, this metric does not include some kinds of dynamically allocated kernel memory. This is now reported in the GBL_MEM_SYS* metrics.

Large fluctuations in this metric can be caused by programs which allocate large amounts of memory and then either release the memory or terminate. A slow continual increase in this metric may indicate a program with a memory leak.

GBL_MEM_UTIL

The percentage of physical memory in use during the interval. This includes system memory (occupied by the kernel), buffer cache and user memory.

On HP-UX 11iv3 and above, this includes file cache also.

On HP-UX, this calculation is done using the byte values for physical memory and used memory, and is therefore more accurate than comparing the reported kilobyte values for physical memory and used memory.

On Linux, the value of this metric includes buffer cache when the cachemem parameter in the parm file is set to user.

On SUN, high values for this metric may not indicate a true memory shortage. This metric can be influenced by the VMM (Virtual Memory Management) system.

Locality Domain metrics are available on HP-UX 11iv2 and above. GBL_MEM_FREE and LDOM_MEM_FREE, as well as the memory utilization metrics derived from them, may not always fully match. GBL_MEM_FREE represents free memory in the kernel's reservation layer while LDOM_MEM_FREE shows actual free pages. If memory has been reserved but not actually consumed from the Locality Domains, the two values won't match. Because GBL_MEM_FREE includes pre-reserved memory, the GBL_MEM_* metrics are a better indicator of actual memory consumption in most situations.

GBL_MEM_UTIL_CUM

The average percentage of physical memory in use over the cumulative collection time. This includes system memory (occupied by the kernel), buffer cache and user memory.

On HP-UX 11iv3 and above, this includes file cache also.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_UTIL_HIGH

The highest percentage of physical memory in use in any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_MEM_VIRT

The total private virtual memory (in MBs unless otherwise specified) at the end of the interval. This is the sum of the virtual allocation of private data and stack regions for all processes.

GBL_MEM_WAIT_PCT

The percentage of time processes or kernel threads were blocked on VM (waiting for virtual memory resources to become available) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on VM divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_MEM_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on VM (waiting for virtual memory resources to become available) during the interval.

GBL_MI_LOST_PROC

The number of processes the measurement layer has lost the ability to update during the interval. This is an indication the system activity might require the midaemon be restarted with a larger process count. See the midaemon man page for additional information on the -pids parameter.

GBL_MI_LOST_PROC_CUM

The total number of processes the measurement layer has lost the ability to update during the cumulative collection interval.

GBL_MI_PROC_ENTRIES

The number of process entries allocated in the midaemon shared memory area.

GBL_MI_THREAD_ENTRIES

The number of thread entries allocated in the midaemon shared memory area.

GBL_MSG_QUEUE

The average number of processes or kernel threads blocked on messages (waiting for their message queue calls to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on MESG (that is, messages) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_MSG_WAIT_PCT

The percentage of time processes or kernel threads were blocked on messages (waiting for their message queue calls to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on MESG (that is, messages) divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_MSG_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on messages (waiting for their message queue calls to complete) during the interval.

GBL_NETWORK_SUBSYSTEM_QUEUE

The average number of processes or kernel threads blocked on the network subsystem (waiting for their network activity to complete) during the interval. This is the sum of processes or kernel threads in the LAN, NFS, and RPC wait states. This does not include processes or kernel threads blocked on SOCKT (that is, sockets) waits, as some processes or kernel threads sit idle in SOCKT waits for long periods.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on (LAN + NFS + RPC) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_NETWORK_SUBSYSTEM_WAIT_PCT

The percentage of time processes or kernel threads were blocked on the network subsystem (waiting for their network activity to complete) during the interval. This is the sum of processes or kernel threads in the LAN, NFS, and RPC wait states. This does not include processes or kernel threads blocked on SOCKT (that is, sockets) waits, as some processes or kernel threads sit idle in SOCKT waits for long periods.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on (LAN + NFS + RPC) divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_NET_COLLISION

The number of collisions that occurred on all network interfaces during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not include deferred packets.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the “Single Collision Frames“, ”Multiple Collision Frames“, ”Late Collisions“, and ”Excessive Collisions“ values from the output of the ”lanadmin“ utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the “Coll” column from the “netstat -i” command (“collisions” from the “netstat -i -e” command on Linux) for a network device. See also netstat(1).

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_COLLISION_1_MIN_RATE

The number of collisions per minute on all network interfaces during the interval. This metric does not include deferred packets.

This does not include data for loopback interface.

Collisions occur on any busy network, but abnormal collision rates could indicate a hardware or software problem.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_COLLISION_CUM

The number of collisions that occurred on all network interfaces over the cumulative collection time. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not include deferred packets.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For HP-UX, this will be the same as the sum of the “Single Collision Frames“, ”Multiple Collision Frames“, ”Late Collisions“, and ”Excessive Collisions“ values from the output of the ”lanadmin“ utility for the network interface. Remember that “lanadmin” reports cumulative counts. For this release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For other Unix systems, this is the same as the sum of the “Coll” column from the “netstat -i” command (“collisions” from the “netstat -i -e” command on Linux) for a network device. See also netstat(1).

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_COLLISION_PCT

The percentage of collisions to total outbound packet attempts during the interval. Outbound packet attempts include both successful packets and collisions.

This does not include data for loopback interface.

A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested.

This metric does not currently include deferred packets.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_COLLISION_PCT_CUM

The percentage of collisions to total outbound packet attempts over the cumulative collection time. Outbound packet attempts include both successful packets and collisions.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested.

This metric does not currently include deferred packets.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_COLLISION_RATE

The number of collisions per second on all network interfaces during the interval. This metric does not include deferred packets.

This does not include data for loopback interface.

A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_DEFERRED

The number of outbound deferred packets due to the network being in use during the interval.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_DEFERRED_CUM

The number of outbound deferred packets due to the network being in use over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_DEFERRED_PCT

The percentage of deferred packets to total outbound packet attempts during the interval. Outbound packet attempts include both packets successfully transmitted and those that were deferred.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_DEFERRED_PCT_CUM

The percentage of deferred packets to total outbound packet attempts over the cumulative collection time. Outbound packet attempts include both packets successfully transmitted and those that were deferred.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_DEFERRED_RATE

The number of deferred packets per second on all network interfaces during the interval.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_DEFERRED_RATE_CUM

The number of deferred packets per second on all network interfaces over the cumulative collection time.

This does not include data for loopback interface. The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_ERROR

The number of errors that occurred on all network interfaces during the interval.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the “Inbound Errors” and “Outbound Errors” values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of “Ierrs” (RX-ERR on Linux) and “Oerrs” (TX-ERR on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_ERROR_1_MIN_RATE

The number of errors per minute on all network interfaces during the interval. This rate should normally be zero or very small. A large error rate can indicate a hardware or software problem.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_ERROR_CUM

The number of errors that occurred on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For HP-UX, this will be the same as the total sum of the “Inbound Errors” and “Outbound Errors” values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of “Ierrs” (RX-ERR on Linux) and “Oerrs” (TX-ERR on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_ERROR_RATE

The number of errors per second on all network interfaces during the interval.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_ERROR

The number of inbound errors that occurred on all network interfaces during the interval.

A large number of errors may indicate a hardware problem on the network.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the “Inbound Errors” values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of “Ierrs” (RX-ERR on Linux) and “Oerrs” (TX-ERR on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_ERROR_CUM

The number of inbound errors that occurred on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

A large number of errors may indicate a hardware problem on the network.

For HP-UX, this will be the same as the total sum of the “Inbound Errors” values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of “Ierrs” (RX-ERR on Linux) and “Oerrs” (TX-ERR on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_ERROR_PCT

The percentage of inbound network errors to total inbound packet attempts during the interval. Inbound packet attempts include both packets successfully received and those that encountered errors.

This does not include data for loopback interface.

A large number of errors may indicate a hardware problem on the network. The percentage of inbound errors to total packets attempted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_ERROR_PCT_CUM

The percentage of inbound network errors to total inbound packet attempts over the cumulative collection time. Inbound packet attempts include both packets successfully received and those that encountered errors.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

A large number of errors may indicate a hardware problem on the network. The percentage of inbound errors to total packets attempted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_ERROR_RATE

The number of inbound errors per second on all network interfaces during the interval.

This does not include data for loopback interface.

A large number of errors may indicate a hardware problem on the network. The percentage of inbound errors to total packets attempted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_ERROR_RATE_CUM

The average number of inbound errors per second on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_PACKET

The number of successful packets received through all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the “Inbound Unicast Packets“ and ”Inbound Non-Unicast Packets“ values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the “Ipkts” column (RX-OK on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_PACKET_CUM

The number of successful packets received through all network interfaces over the cumulative collection time. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For HP-UX, this will be the same as the total sum of the “Inbound Unicast Packets“ and ”Inbound Non-Unicast Packets“ values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the “Ipkts” column (RX-OK on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_PACKET_RATE

The number of successful packets per second received through all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IP_FRAGMENTS_RECEIVED

The number of valid IPv4 datagram fragments received by the host.

GBL_NET_IP_FWD_DATAGRAMS

The number of IPv4 datagrams this host has forwarded. In other words, the number of IPv4 datagrams for which this host has been used as a router.

GBL_NET_IP_REASSEMBLY_REQUIRED

The number of IPv4 datagram fragments sent to this host for local delivery which required reassembly before being given to the Upper Layer Protocol(s).

GBL_NET_OUTQUEUE

The sum of the outbound queue lengths for all network interfaces (BYNETIF_QUEUE). This metric is derived from the same source as the Outbound Queue Length shown in the lanadmin(1M) program.

This does not include data for loopback interface.

For most interfaces, the outbound queue is usually zero. When the value is non-zero over a period of time, the network may be experiencing a bottleneck. Determine which network interface has a non-zero queue and compare its traffic levels to normal. Also see if processes are blocking on network wait states.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_ERROR

The number of outbound errors that occurred on all network interfaces during the interval.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the “Outbound Errors” values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of “Oerrs” (TX-ERR on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_ERROR_CUM

The number of outbound errors that occurred on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For HP-UX, this will be the same as the total sum of the “Outbound Errors” values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of “Oerrs” (TX-ERR on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_ERROR_PCT

The percentage of outbound network errors to total outbound packet attempts during the interval. Outbound packet attempts include both packets successfully sent and those that encountered errors.

This does not include data for loopback interface.

The percentage of outbound errors to total packets attempted to be transmitted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_OUT_ERROR_PCT_CUM

The percentage of outbound network errors to total outbound packet attempts over the cumulative collection time. Outbound packet attempts include both packets successfully sent and those that encountered errors.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

The percentage of outbound errors to total packets attempted to be transmitted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_ERROR_RATE

The number of outbound errors per second on all network interfaces during the interval.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_OUT_ERROR_RATE_CUM

The number of outbound errors per second on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_PACKET

The number of successful packets sent through all network interfaces during the last interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the “Outbound Unicast Packets“ and ”Outbound Non-Unicast Packets“ values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the “Opkts” column (TX-OK on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_OUT_PACKET_CUM

The number of successful packets sent through all network interfaces over the cumulative collection time. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For HP-UX, this will be the same as the total sum of the “Outbound Unicast Packets“ and ”Outbound Non-Unicast Packets“ values from the output of the “lanadmin” utility for the network interface. Remember that “lanadmin” reports cumulative counts. As of the HP-UX 11.0 release and beyond, “netstat -i” shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the “Opkts” column (TX-OK on Linux) from the “netstat -i” command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_PACKET_RATE

The number of successful packets per second sent through the network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_PACKET

The total number of successful inbound and outbound packets for all network interfaces during the interval. These are the packets that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

GBL_NET_PACKET_RATE

The number of successful packets per second (both inbound and outbound) for all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_UTIL_PEAK

It is the utilisation of the most used network interfaces at the end of the interval.

GBL_NFS_CALL

The number of NFS calls the local system has made as either a NFS client or server during the interval.

This includes both successful and unsuccessful calls. Unsuccessful calls are those that cannot be completed due to resource limitations or LAN packet errors.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

On AIX System WPARs, this metric is NA.

GBL_NFS_CALL_RATE

The number of NFS calls per second the system made as either a NFS client or NFS server during the interval.

Each computer can operate as both a NFS server, and as an NFS client.

This metric includes both successful and unsuccessful calls. Unsuccessful calls are those that cannot be completed due to resource limitations or LAN packeterrors.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

On AIX System WPARs, this metric is NA.

GBL_NFS_CLIENT_BAD_CALL

The number of failed NFS client calls during the interval. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

GBL_NFS_CLIENT_BAD_CALL_CUM

The number of failed NFS client calls over the cumulative collection time. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_CLIENT_BIOD

The current number of biods running (both idle and active) at the end of the interval.

GBL_NFS_CLIENT_BYTE

The total number of KBs the local machine has sent or received as an NFS client during the interval.

Each computer can operate as both an NFS server, and as a NFS client.

GBL_NFS_CLIENT_BYTE_CUM

The total number of KBs the local machine has sent or received as an NFS client over the cumulative collection time.

Each computer can operate as both an NFS server, and as a NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_CLIENT_CALL

The number of NFS calls the local machine has processed as a NFS client during the interval. Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate should exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

GBL_NFS_CLIENT_CALL_CUM

The number of NFS calls the local machine has processed as a NFS client over the cumulative collection time. Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate should exceed the IO rate. This metric includes both successful and unsuccessful calls.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

GBL_NFS_CLIENT_CALL_RATE

The number of NFS calls the local machine has processed as a NFS client per second during the interval. Calls are the system call used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate should exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

GBL_NFS_CLIENT_IDLE_BIOD

The current number of biods inactive at the end of the interval. A value of zero indicates a potential bottleneck for the NFS client.

GBL_NFS_CLIENT_IO

The number of NFS IOs the local machine has completed as an NFS client during the interval. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_CLIENT_IO_CUM

The number of NFS IOs the local machine has completed as an NFS client over the cumulative collection time. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_CLIENT_IO_PCT

The percentage of NFs IOs the local machine has completed as an NFS client versus total NFS IOs completed during the interval. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

A percentage greater than 50 indicates that this machine is acting more as a client. A percentage less than 50 indicates this machine is acting more as a server for others.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_CLIENT_IO_PCT_CUM

The percentage of NFS IOs the local machine has completed as an NFS client versus total NFS IOs completed over the cumulative collection time. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

A percentage greater than 50 indicates that this machine is acting more as a client. A percentage less than 50 indicates this machine is acting more as a server for others.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_CLIENT_IO_RATE

The number of NFS IOs per second the local machine has completed as an NFS client during the interval. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_CLIENT_IO_RATE_CUM

The number of NFS IOs per second the local machine has completed as an NFS client over the cumulative collection time. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_CLIENT_PHYS_TIME

The time, in seconds, spent to service all NFS operations as a NFS client during the last interval. This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the “service time” less the local machine's software overhead.

GBL_NFS_CLIENT_PHYS_TIME_CUM

The time, in seconds, spent to service all NFS operations as a NFS client over the cumulative collection time. This is measured from the time the operation gets onto the physical network until the time a reply is received from the network. In other words, this is the “service time” less the local machine's software overhead.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_CLIENT_READ_BYTE_RATE

The number of KBs per second the system received as an NFS client doing read operations during the interval.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_CLIENT_READ_BYTE_RATE_CUM

The average number of KBs per second the system received as an NFS client doing read operations over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_CLIENT_READ_RATE

The number of NFS “read” operations per second the system generated as an NFS client during the interval.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_CLIENT_READ_RATE_CUM

The average number of NFS “read” operations per second the system generated as an NFS client over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_CLIENT_SERVICE_QUEUE

The number of pending NFS client operations during the interval. This value increases as the service time increases and/or as the rate of client requests increases.

GBL_NFS_CLIENT_SERVICE_QUEUE_CUM

The average number of pending NFS client operations per interval over the cumulative collection time. Queue length increases as the service time increases and/or as the rate of client requests increases.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_CLIENT_SERVICE_TIME

The time, in seconds, spent to service all NFS operations as a NFS client during the last interval. This is the time from the point that the client originates the requests to the point replies are received including IO buffering, NFS and network software layer delays, physical network latency, and NFS server service time. It is not a measure of the average response time per NFS request. This can be thought of as the round-trip time for all NFS requests made during the interval.

GBL_NFS_CLIENT_SERVICE_TIME_CUM

The time, in seconds, spent to service all NFS operations as a NFS client over the cumulative collection time. This is the time from the point that the client originates the request to the point a reply is received including IO buffering, NFS and network software layer delays, physical network latency, and NFS server service time. It is not a measure of the average response time per nfs request. This can be thought of as the round-trip time for all nfs requests.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_CLIENT_WRITE_BYTE_RATE

The number of KBs per second the system sent over the network as an NFS client doing write operations during the interval.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_CLIENT_WRITE_BYTE_RATE_CUM

The average number of KBs per second the system sent over the network as an NFS client doing write operations over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_CLIENT_WRITE_RATE

The number of NFS “write” operations per second the system generated as an NFS client during the interval.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_CLIENT_WRITE_RATE_CUM

The average number of NFS “write” operations per second the system generated as an NFS client over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_LOGL_READ

The number of logical reads made to NFS disks by the local machine as a NFS client during the interval.

Each computer can operate as both a NFS server, and as an NFS client. For this metric the local machine is acting as a NFS client (that is, the disks are remote) since if it were acting as a server the logical disk requests would be going to local disks. These logical requests do not necessarily result in a physical IO request across the NFS link.

GBL_NFS_LOGL_READ_BYTE

The number of KBs transferred through logical reads to NFS disks by the local machine during the interval. Note that these are transfers by read calls, not physical IO.

GBL_NFS_LOGL_READ_BYTE_CUM

The number of KBs transferred through logical reads to NFS disks by the local machine over the cumulative collection time. Note that these are transfers by read calls, not physical IO.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_LOGL_READ_CUM

The total number of logical reads made to NFS disks by the local machine as a NFS client over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Each computer can operate as both a NFS server, and as an NFS client. For this metric the local machine is acting as an NFS client (the disks are remote) since if it were acting as a server the logical disk requests would be going to local disks. These logical requests do not necessarily result in a physical IO request across the NFS link.

GBL_NFS_LOGL_READ_PCT

The percentage of logical reads to total logical reads and writes to NFS disks by the local machine during the interval.

GBL_NFS_LOGL_READ_PCT_CUM

The average percentage of logical reads to total logical reads and writes to NFS disks by the local machine over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_LOGL_READ_RATE

The number of logical reads per second made to NFS disks by the local machine during the interval.

GBL_NFS_LOGL_READ_RATE_CUM

The average number of logical reads per second made to NFS disks by the local machine over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_LOGL_WRITE

The number of logical writes made to NFS disks by the local machine during the interval.

Each computer can operate as both a NFS server, and as a NFS client. For this metric the local machine is acting as an NFS client (the disks are remote) since if it were acting as a server the logical disk requests would be going to local disks. These logical requests do not necessarily result in a physical IO request across the NFS link.

GBL_NFS_LOGL_WRITE_BYTE

The number of KBs transferred through logical writes to NFS disks by the local machine during the interval. Note that these are transfers by write calls, not physical IO.

GBL_NFS_LOGL_WRITE_BYTE_CUM

The number of KBs transferred through logical writes to NFS disks by the local machine over the cumulative collection time. Note that these are transfers by write calls, not physical IO.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_LOGL_WRITE_CUM

The total number of logical writes made to NFS disks by the local machine over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Each computer can operate as both a NFS server, and as a NFS client. For this metric the local machine is acting as an NFS client (the disks are remote) since if it were acting as a server the logical disk requests would be going to local disks. These logical requests do not necessarily result in a physical IO request across the NFS link.

GBL_NFS_LOGL_WRITE_PCT

The percentage of logical writes to total logical reads and writes to NFS disks by the local machine during the interval.

GBL_NFS_LOGL_WRITE_PCT_CUM

The average percentage of logical writes to total logical IO to NFS disks by the local machine over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_LOGL_WRITE_RATE

The number of logical writes per second made to NFS disks by the local machine during the interval.

GBL_NFS_LOGL_WRITE_RATE_CUM

The average number of logical writes per second made to NFS disks by the local machine over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_QUEUE

The average number of processes or kernel threads blocked on NFS (waiting for their network file system IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on NFS divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_NFS_SERVER_BAD_CALL

The number of failed NFS server calls during the interval. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

GBL_NFS_SERVER_BAD_CALL_CUM

The number of failed NFS server calls over the cumulative collection time. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_SERVER_BYTE

The number of KBs the local machine has processed as a NFS server during the interval.

Each computer can operate as both a NFS server, and as an NFS client.

GBL_NFS_SERVER_BYTE_CUM

The number of KBs the local machine has processed as a NFS server over the cumulative collection time.

Each computer can operate as both a NFS server, and as an NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_SERVER_CALL

The number of NFS calls the local machine has processed as a NFS server during the interval.

Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate could exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

GBL_NFS_SERVER_CALL_CUM

The number of NFS calls the local machine has processed as a NFS server over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate could exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

GBL_NFS_SERVER_CALL_RATE

The number of NFS calls the local machine has processed per second as a NFS server during the interval.

Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate could exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

GBL_NFS_SERVER_IO

The number of NFS IOs the local machine has completed as an NFS server during the interval. This number represents physical IOs received by the serverein contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_SERVER_IO_CUM

The number of NFS IOs the local machine has completed as an NFS server over the cumulative collection time. This number represents physical IOs received by the server n contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_SERVER_IO_PCT

The percentage of NFS IOs the local machine has completed as an NFS server versus total NFS IOs completed during the interval. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

A percentage greater than 50 indicates that this machine is acting more as a server for others. A percentage less than 50 indicates this machine is acting more as a client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_SERVER_IO_PCT_CUM

The percentage of NFs IOs the local machine has completed as an NFS server versus total NFS IOs completed over the cumulative collection time. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

A percentage greater than 50 indicates that this machine is acting more as a server for others. A percentage less than 50 indicates this machine is acting more as a client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_SERVER_IO_RATE

The number of NFS IOs per second the local machine has completed as an NFS server during the interval. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_SERVER_IO_RATE_CUM

The number of NFS IOs per second the local machine has completed as an NFS server over the cumulative collection time. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writecache.

GBL_NFS_SERVER_READ_BYTE_RATE

The number of KBs per second the system sent as a NFS server responding to NFS read operations from client nodes during the interval.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_SERVER_READ_BYTE_RATE_CUM

The average number of KBs per second the system sent as an NFS server responding to NFS read operations from client nodes over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_SERVER_READ_RATE

The number of NFS “read” operations per second the system processed as an NFS server during the interval.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_SERVER_READ_RATE_CUM

The average number of NFS “read” operations per second the system processed as an NFS server over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_SERVER_SERVICE_TIME

The time, in seconds, spent for the NFS server to process the client's operations during the interval. This includes all of the time from the point that the operations are received to the point where a reply is sent back to the client, which includes software overhead and any local disk IOs. This is not an average service time per operation; it is the total service time for all operations processed during the interval.

GBL_NFS_SERVER_SERVICE_TIME_CUM

The time, in seconds, spent for the NFS server to process the client's operations over the cumulative collection time. This includes all of the time from the point that the operations are received to the point where a reply is sent back to the client, which includes software overhead and any local disk IOs. This is not an average service time per operation; it is the total service time for all operations processed.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_NFS_SERVER_WRITE_BYTE_RATE

The number of KBs per second the system received over the network as an NFS server performing write operations for client nodes during the interval.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_SERVER_WRITE_BYTE_RATE_CUM

The average number of KBs per second the system received over the network as an NFS server performing write operations for client nodes over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_SERVER_WRITE_RATE

The number of NFS “write” operations per second the system processed as an NFS server during the interval.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_SERVER_WRITE_RATE_CUM

The average number of NFS “write” operations per second the system processed as an NFS server over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

NFS Version 2 write operations consist of setattr, write, writecache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_WAIT_PCT

The percentage of time processes or kernel threads were blocked on NFS (waiting for their network file system IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on NFS divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_NFS_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on NFS (waiting for their network file system IO to complete) during the interval.

GBL_NODENAME

On Unix systems, this is the name of the computer as returned by the command “uname -n” (that is, the string returned from the “hostname” program).

On Windows, this is the name of the computer as returned by GetComputerName.

GBL_NUM_ACTIVE_LS

This indicates the number of LS hosted in a system that are active . If Perf Agent is installed in a guest or in a standalone system this value will be 0.

On Solaris non-global zones, this metric shows value as 0.

GBL_NUM_APP

The number of applications defined in the parm file plus one (for “other”).

The application called “other” captures all other processes not defined in the parm file.

You can define up to 999 applications.

GBL_NUM_APP_PRM

The number of PRM groups configured - 1 per PRM Group ID. HP-UX supports up to 64 unique PRM Groups.

GBL_NUM_CPU

The number of physical CPUs on the system. This includes all CPUs, either online or offline. For HP-UX and certain versions of Linux, the sar(1M) command allows you to check the status of the system CPUs. For SUN and DEC, the commands psrinfo(1M) and psradm(1M) allow you to check or change the status of the system CPUs. For AIX, this metric indicates the maximum number of CPUs the system ever had.

On a logical system, this metric indicates the number of virtual CPUs configured. When hardware threads are enabled, this metric indicates the number of logical processors.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be “na”, some will be based on what is visible at startup time. All information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if

CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

GBL_NUM_CPU_CORE

This metric provides the total number of CPU cores on a physical system. On VMs, this metric shows information according to resources available on that VM. On non HP-UX system, this metric is equivalent to active CPU cores. On AIX System WPARs, this metric value is identical to the value on AIX Global Environment. On Windows, this metric will be “na” on Windows Server 2003 Itanium systems.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be “na”, some will be based on what is visible at startup time. All information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

GBL_NUM_DISK

The number of disks on the system. Only local disk devices are counted in this metric.

On HP-UX, this is a count of the number of disks on the system that have ever had activity over the cumulative collection time.

On Solaris non-global zones, this metric shows value as 0.

On AIX System WPARs, this metric shows value as 0.

GBL_NUM_HBA

The number of Host Bus adaptors on the system. This metric is supported on HP-UX 11iv3 and above.

GBL_NUM_LDOM

The number of active Locality Domains in the system.

GBL_NUM_LS

This indicates the number of LS hosted in a system. If Perf Agent is installed in a guest or in a standalone system this value will be 0.

On Solaris non-global zones, this metric shows value as 0.

GBL_NUM_NETWORK

The number of network interfaces on the system. This includes the loopback interface. On certain platforms, this also include FDDI, Hyperfabric, ATM, Serial Software interfaces such as SLIP or PPP, and Wide Area Network interfaces (WAN) such as ISDN or X.25. The “netstat -i” command also displays the list of network interfaces on the system.

GBL_NUM_SOCKET

The number of physical cpu sockets on the system. On VMs, this metric shows information according to resources available on that VM.

On Windows, this metric will be “na” on Windows Server 2003 Itanium systems.

GBL_NUM_SWAP

The number of configured swap areas.

GBL_NUM_TAPE

The number of Tape devices attached to the system. This metric is supported on HP-UX 11iv3 and above.

GBL_NUM_TT

The number of unique Transaction Tracker (TT) transactions that have been registered on this system.

GBL_NUM_USER

The number of users logged in at the time of the interval sample. This is the same as the command “who | wc -l”.

For Unix systems, the information for this metric comes from the utmp file which is updated by the login command. For more information, read the man page for utmp. Some applications may create users on the system without using login and updating the utmp file. These users are not reflected in this count.

This metric can be a general indicator of system usage. In a networked environment, however, users may maintain inactive logins on several systems.

On Windows, the information for this metric comes from the Server Sessions counter in the Performance Libraries Server object. It is a count of the number of users using this machine as a file server.

GBL_NUM_VG

The number of available volume groups.

GBL_NUM_VSWITCH

The number of virtual switches configured on the host system.

GBL_OSKERNELTYPE

This indicates the word size of the current kernel on the system. Some hardware can load the 64-bit kernel or the 32-bit kernel.

GBL_OSKERNELTYPE_INT

This indicates the word size of the current kernel on the system. Some hardware can load the 64-bit kernel or the 32-bit kernel.

GBL_OSNAME

A string representing the name of the operating system. On Unix systems, this is the same as the output from the “uname -s” command.

GBL_OSRELEASE

The current release of the operating system.

On most Unix systems, this is same as the output from the “uname -r” command.

On AIX, this is the actual patch level of the operating system. This is similar to what is returned by the command “lslpp -l bos.rte” as the most recent level of the COMMITTED Base OS Runtime. For example, “5.2.0”.

GBL_OSVERSION

A string representing the version of the operating system. This is the same as the output from the “uname -v” command. This string is limited to 20 characters, and as a result, the complete version name might be truncated.

On Windows, this is a string representing the service pack installed on the operating system.

GBL_OTHER_IO_QUEUE

The average number of processes or kernel threads blocked on “other IO” during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on other IO divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_OTHER_IO_WAIT_PCT

The percentage of time processes or kernel threads were blocked on “other IO” during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on other IO divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_OTHER_IO_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on “other IO” during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

GBL_OTHER_QUEUE

The average number of processes or kernel threads blocked on other (unknown) activities during the interval. This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on OTHER divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_OTHER_WAIT_PCT

The percentage of time processes or kernel threads were blocked on other (unknown) activities during the interval. This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on OTHER divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_OTHER_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on other (unknown) activities during the interval. This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

GBL_PIPE_QUEUE

The average number of processes or kernel threads blocked onPIPE (waiting for pipe communication to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on PIPE divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_PIPE_WAIT_PCT

The percentage of time processes or kernel threads were blocked onPIPE (waiting for pipe communication to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on PIPE divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_PIPE_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked onPIPE (waiting for pipe communication to complete) during the interval.

GBL_PRI_QUEUE

The average number of processes or kernel threads blocked on PRI (waiting for their priority to become high enough to get the CPU) during the interval.

To determine if the CPU is a bottleneck, compare this metric with GBL_CPU_TOTAL_UTIL. If GBL_CPU_TOTAL_UTIL is near 100 percent and GBL_PRI_QUEUE is greater than three, there is a high probability of a CPU bottleneck.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on PRI divided by the interval time.

HP-UX RUN/PRI/CPU Queue differences for multi-cpu systems:

For example, let's assume we're using a system with eight processors. We start eight CPU intensive threads that consume almost all of the CPU resources. The approximate values shown for the CPU related queue metrics would be:

```
GBL_RUN_QUEUE = 1.0
GBL_PRI_QUEUE = 0.1
GBL_CPU_QUEUE = 1.0
```

Assume we start an additional eight CPU intensive threads. The approximate values now shown are:

```
GBL_RUN_QUEUE = 2.0
GBL_PRI_QUEUE = 8.0
GBL_CPU_QUEUE = 16.0
```

At this point, we have sixteen CPU intensive threads running on the eight processors. Keeping the definitions of the three queue metrics in mind, the run queue is 2 (that is, 16 / 8); the pri queue is 8 (only half of the threads can be active at any given time); and the cpu queue is 16 (half of the threads waiting in the cpu queue that are ready to run, plus one for each active thread).

This illustrates that the run queue is the average of number of threads waiting in the runqueue for all processors; the pri queue is the number of threads that are blocked on “PRI” (priority); and the cpu queue is the number of threads in the cpu queue that are ready to run, including the threads using the CPU.

Note that if the value for GBL_PRI_QUEUE greatly exceeds the value for GBL_RUN_QUEUE, this may be a side-effect of the measurement interface having lost trace data. In this case, check the value of the GBL_LOST_MI_TRACE_BUFFERS metric. If there has been buffer loss, you can correct the value of GBL_PRI_QUEUE by restarting the midaemon and the performance tools. You can use the /opt/perf/bin/midaemon -T command to force immediate shutdown of the measurement interface.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_PRI_WAIT_PCT

The percentage of time processes or kernel threads were blocked on PRI (waiting for their priority to become high enough to get the CPU) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on PRI divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_ SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_PRI_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on PRI (waiting for their priority to become high enough to get the CPU) during the interval.

GBL_PRM_MEM_UTIL

The total percent of memory used by processes within the PRM groups during the interval. This does not include system processes (processes attached to PRM group 0).

GBL_PROC_RUN_TIME

The average run time, in seconds, for processes that terminated during the interval.

GBL_PROC_SAMPLE

The number of process data samples that have been averaged into global metrics (such as GBL_ ACTIVE_PROC) that are based on process samples.

GBL_RPC_QUEUE

The average number of processes or kernel threads blocked on RPC (waiting for their remote procedure calls to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on RPC divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_RPC_WAIT_PCT

The percentage of time processes or kernel threads were blocked on RPC (waiting for their remote procedure calls to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on RPC divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_RPC_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on RPC (waiting for their remote procedure calls to complete) during the interval.

GBL_RUN_QUEUE

On UNIX systems except Linux, this is the average number of threads waiting in the runqueue over the interval. The average is computed against the number of times the run queue is occupied instead of time. The average is updated by the kernel at a fine grain interval, only when the run queue is occupied. It is not averaged against the interval and can therefore be misleading for long intervals when the run queue is empty most or part of the time. This value matches runq-sz reported by the “sar -q” command. The GBL_LOADAVG* metrics are better indicators of run queue pressure.

On Linux and Windows, this is instantaneous value obtained at the time of logging. On Linux, it shows the number of threads waiting in the runqueue. On Windows, it shows the Processor Queue Length.

On Unix systems, GBL_RUN_QUEUE will typically be a small number. Larger than normal values for this metric indicate CPU contention among threads. This CPU bottleneck is also normally indicated by 100 percent GBL_CPU_TOTAL_UTIL. It may be OK to have GBL_CPU_TOTAL_UTIL be 100 percent if no other threads are waiting for the CPU. However, if GBL_CPU_TOTAL_UTIL is 100 percent and GBL_RUN_QUEUE is greater than the number of processors, it indicates a CPU bottleneck.

On Windows, the Processor Queue reflects a count of process threads which are ready to execute. A thread is ready to execute (in the Ready state) when the only resource it is waiting on is the processor. The Windows operating system itself has many system threads which intermittently use small amounts of processor time. Several low priority threads intermittently wake up and execute for very short intervals. Depending on when the collection process samples this queue, there may be none or several of these low-priority threads trying to execute. Therefore, even on an otherwise quiescent system, the Processor Queue Length can be high. High values for this metric during intervals where the overall CPU utilization (gbl_cpu_total_util) is low do not indicate a performance bottleneck. Relatively high values for this metric during intervals where the overall CPU utilization is near 100% can indicate a CPU performance bottleneck.

HP-UX RUN/PRI/CPU Queue differences for multi-cpu systems:

For example, let's assume we're using a system with eight processors. We start eight CPU intensive threads that consume almost all of the CPU resources. The approximate values shown for the CPU related queue metrics would be:

```
GBL_RUN_QUEUE = 1.0
GBL_PRI_QUEUE = 0.1
GBL_CPU_QUEUE = 1.0
```

Assume we start an additional eight CPU intensive threads. The approximate values now shown are:

```
GBL_RUN_QUEUE = 2.0
GBL_PRI_QUEUE = 8.0
GBL_CPU_QUEUE = 16.0
```

At this point, we have sixteen CPU intensive threads running on the eight processors. Keeping the definitions of the three queue metrics in mind, the run queue is 2 (that is, 16 / 8); the pri queue is 8 (only half of the threads can be active at any given time); and the cpu queue is 16 (half of the threads waiting in the cpu queue that are ready to run, plus one for each active thread).

This illustrates that the run queue is the average of number of threads waiting in the runqueue for all processors; the pri queue is the number of threads that are blocked on “PRI” (priority); and the cpu queue is the number of threads in the cpu queue that are ready to run, including the threads using the CPU.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_RUN_QUEUE_CUM

On UNIX systems except Linux, this is the average number of threads waiting in the runqueue over the cumulative collection time.

On Linux, this is approximately the number of threads waiting in the runqueue over the cumulative collection time.

On Windows, this is approximately the average Processor Queue Length over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

In this case, this metric is a cumulative average of data that was collected as an average. This metric is derived from GBL_RUN_QUEUE.

HP-UX RUN/PRI/CPU Queue differences for multi-cpu systems:

For example, let's assume we're using a system with eight processors. We start eight CPU intensive threads that consume almost all of the CPU resources. The approximate values shown for the CPU related queue metrics would be:

```
GBL_RUN_QUEUE = 1.0
GBL_PRI_QUEUE = 0.1
GBL_CPU_QUEUE = 1.0
```

Assume we start an additional eight CPU intensive threads. The approximate values now shown are:

```
GBL_RUN_QUEUE = 2.0
GBL_PRI_QUEUE = 8.0
GBL_CPU_QUEUE = 16.0
```

At this point, we have sixteen CPU intensive threads running on the eight processors. Keeping the definitions of the three queue metrics in mind, the run queue is 2 (that is, 16 / 8); the pri queue is 8 (only half of the threads can be active at any given time); and the cpu queue is 16 (half of the threads waiting in the cpu queue that are ready to run, plus one for each active thread).

This illustrates that the run queue is the average of number of threads waiting in the runqueue for all processors; the pri queue is the number of threads that are blocked on “PRI” (priority); and the cpu queue is the number of threads in the cpu queue that are ready to run, including the threads using the CPU.

GBL_RUN_QUEUE_HIGH

On UNIX systems except Linux, this is the highest value of average number of threads waiting in the runqueue during any interval over the cumulative collection time.

On Linux, this is the highest value of number of threads waiting in the runqueue during any interval over the cumulative collection time.

GBL_SAMPLE

The number of data samples (intervals) that have occurred over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

GBL_SEM_QUEUE

The average number of processes or kernel threads blocked onsemaphores (waiting for their semaphore operations to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on PRI (that is, priority) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SEM_WAIT_PCT

The percentage of time processes or kernel threads were blocked onsemaphores (waiting on a semaphore operation) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on SEM (that is, semaphores) divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the

system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SEM_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked onsemaphores (waiting for their semaphore operations to complete) during the interval.

GBL_SERIALNO

On HP-UX, this is the ID number of the computer as returned by the command “uname -i”. If this value is not available, an empty string is returned.

On SUN, this is the ASCII representation of the hardware-specific serial number. This is printed in hexadecimal as presented by the “hostid” command when possible. If that is not possible, the decimal format is provided instead.

On AIX, this is the machine ID number as returned by the command “uname -m”. This number has the form xxyyyyyymmss. For the RISC System/6000, “xx” position is always 00. The “yyyyyy” positions contain the unique ID number for the central processing unit (cpu). While “mm” represents the model number, and “ss” is the submodel number (always 00).

On Linux, this is the ASCII representation of the hardware-specific serial number, as returned by the command “hostid”.

GBL_SLEEP_QUEUE

The average number of processes or kernel threads blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on SLEEP divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the

system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SLEEP_WAIT_PCT

The percentage of time processes or kernel threads were blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on SLEEP divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SLEEP_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

GBL_SOCKET_QUEUE

The average number of processes or kernel threads blocked on sockets (waiting for their IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on SOCKT (that is, sockets) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SOCKET_WAIT_PCT

The percentage of time processes or kernel threads were blocked on sockets (waiting for their IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on SOCKT (that is, sockets) divided by the accumulated time that all processes or threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SOCKET_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on sockets (waiting for their IO to complete) during the interval.

GBL_STARTDATE

The date that the collector started.

GBL_STARTED_PROC

The number of processes that started during the interval.

GBL_STARTED_PROC_RATE

The number of processes that started per second during the interval.

GBL_STARTTIME

The time of day that the collector started.

GBL_STATDATE

The date at the end of the interval, based on local time.

GBL_STATTIME

An ASCII string representing the time at the end of the interval, based on local time.

GBL_STREAM_QUEUE

The average number of processes or kernel threads blocked on streams IO (waiting for a streams IO operation to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on STRMS (that is, streams IO) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_STREAM_WAIT_PCT

The percentage of time processes or kernel threads were blocked on streams IO (waiting for a streams IO operation to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on STRMS (that is, streams IO) divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_STREAM_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on streams IO (waiting for a streams IO operation to complete) during the interval.

GBL_SWAP_RESERVED_ONLY_UTIL

The percentage of available swap space reserved (for currently running programs), but not yet used.

Swap space must be reserved (but not allocated) before virtual memory can be created. Swap space locations are actually assigned (used) when a page is actually written to disk.

On HP-UX, when compared to the “swapinfo -mt” command results, this is calculated as:

```
  Util = ((USED: reserve)
     / (AVAIL: total)) * 100
```

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_AVAIL

The total amount of potential swap space, in MB.

On HP-UX, this is the sum of the device swap areas enabled by the swapon command, the allocated size of any file system swap areas, and the allocated size of pseudo swap in memory if enabled. Note that this is potential swap space. This is the same as (AVAIL: total) as reported by the “swapinfo -mt” command.

On SUN, this is the total amount of swap space available from the physical backing store devices (disks) plus the amount currently available from main memory. This is the same as (used + available) /1024, reported by the “swap -s” command.

On Linux, this is same as (Swap: total) as reported by the “free -m” command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_SWAP_SPACE_AVAIL_KB

The total amount of potential swap space, in KB.

On HP-UX, this is the sum of the device swap areas enabled by the swapon command, the allocated size of any file system swap areas, and the allocated size of pseudo swap in memory if enabled. Note that this is potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space.

On HP-UX, this is the same as (AVAIL: total) as reported by the “swapinfo -t” command.

On SUN, this is the total amount of swap space available from the physical backing store devices (disks) plus the amount currently available from main memory. This is the same as (used + available)/1024, reported by the “swap -s” command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_SWAP_SPACE_DEVICE_UTIL

On HP-UX, this is the percentage of device swap space currently in use of the total swap space available. This does not include file system or remote swap space.

On HP-UX, note that available swap is only potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space.

Consequently, 100 percent utilization on a single device is not always obtainable. The wasted swap space, and the remainder of allocated SWCHUNKs that have not been used is what is reported in the hold field of the /usr/sbin/swapinfo command.

On HP-UX, when compared to the “swapinfo -mt” command results, this is calculated as:

```
  Util = ((USED: dev) sum
    / (AVAIL: total)) * 100
```

On SUN, this is the percentage of total system device swap space currently in use. This metric only gives the percentage of swap space used from the available physical swap device space, and does not include the memory that can be used for swap. (On SunOS 5.X, the virtual swap swapfs can allocate swap space from memory.)

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_FS_UTIL

On HP-UX, this is the percentage file system swap space currently in use of the total swap space available. This includes both local and NFS file system swap. Since file system swap is dynamic (it grows in SWCHUNK sizes as needed and is not bounded as device swap is), this number fluctuates as more swap is allocated.

When compared to the “swapinfo -mt” command results, this is calculated as:

```
  Util = ((USED: fs) sum
    / (AVAIL: total)) * 100
```

On Sinix, this is the percentage of swap space in use of the total swap space provided on regular files that were configured for swap.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

GBL_SWAP_SPACE_RESERVED

The amount of swap space (in MB) reserved for the swapping and paging of programs currently executing. Process pages swapped include data (heap and stack pages), bss (data uninitialized at the beginning of process execution), and the process user area (uarea). Shared memory regions also require the reservation of swap space.

Swap space is reserved (by decrementing a counter) when virtual memory for a program is created, but swap is only used when a page or swap to disk is actually done or the page is locked in memory if swapping to memory is enabled. Virtual memory cannot be created if swap space cannot be reserved.

On HP-UX, this is the same as (USED: total) as reported by the “swapinfo -mt” command.

On SUN, this is the same as used/1024, reported by the “swap -s” command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_RESERVED_UTIL

This is the percentage of available swap space currently reserved for running processes.

Reserved utilization = (amount of swap space reserved / amount of swap space available) * 100

On HP-UX, swap space must be reserved (but not allocated) before virtual memory can be created. If all of available swap is reserved, then no new processes or virtual memory can be created. Swap space locations are actually assigned (used) when a page is actually written to disk.

On HP-UX, note that available swap is only potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space. Consequently, 100 percent utilization on a single device is not always obtainable.

When compared to the “swapinfo -mt” command results, this is calculated as:

```
Util = ((USED: total)
  / (AVAIL: total)) * 100
```

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_USED

The amount of swap space used, in MB.

On HP-UX, “Used” indicates written to disk (or locked in memory), rather than reserved. This is the same as (USED: total - reserve) as reported by the “swapinfo -mt” command.

On SUN, “Used” indicates amount written to disk (or locked in memory), rather than reserved. Swap space is reserved (by decrementing a counter) when virtual memory for a program is created. This is the same as (bytes allocated)/1024, reported by the “swap -s” command.

On Linux, this is same as (Swap: used) as reported by the “free -m” command.

On AIX System WPARs, this metric is NA.

On Solaris non-global zones, this metric is N/A. On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

GBL_SWAP_SPACE_USED_UTIL

This is the percentage of swap space used.

On HP-UX, “Used %” indicates percentage of swap space written to disk (or locked in memory), rather than reserved. This is the same as percentage of ((USED: total - reserve)/total)*100, as reported by the “swapinfo -mt” command.

On SUN, “Used %” indicates percentage of swap space written to disk (or locked in memory), rather than reserved. Swap space is reserved (by decrementing a counter) when virtual memory for a program is created. This is the same as percentage of ((bytes allocated)/total)*100, reported by the “swap -s” command.

On SUN, global swap space is tracked through the operating system. Device swap space is tracked through the devices. For this reason, the amount of swap space used may differ between the global and by-device metrics. Sometimes pages that are marked to be swapped to disk by the operating system are never swapped. The operating system records this as used swap space, but the devices do not, since no physical IOs occur. (Metrics with the prefix “GBL” are global and metrics with the prefix “BYSWP” are by device.)

On Linux, this is same as percentage of ((Swap: used)/total)*100, as reported by the “free -m” command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_UTIL

The percent of available swap space that was being used by running processes in the interval.

On Windows, this is the percentage of virtual memory, which is available to user processes, that is in use at the end of the interval. It is not an average over the entire interval. It reflects the ratio of committed memory to the current commit limit. The limit may be increased by the operating system if the paging file is extended. This is the same as (Committed Bytes / Commit Limit) * 100 when comparing the results to Performance Monitor.

On HP-UX, swap space must be reserved (but not allocated) before virtual memory can be created. If all of available swap is reserved, then no new processes or virtual memory can be created. Swap space locations are actually assigned (used) when a page is actually written to disk or locked in memory (pseudo swap in memory). This is the same as (PCT USED: total) as reported by the “swapinfo -mt” command.

On Unix systems, this metric is a measure of capacity rather than performance. As this metric nears 100 percent, processes are not able to allocate any more memory and new processes may not be able to run. Very low swap utilization values may indicate that too much area has been allocated to swap, and better use of disk space could be made by reallocating some swap partitions to be user filesystems.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_SWAP_SPACE_UTIL_CUM

The average percentage of available swap space currently in use (has memory belonging to processes paged or swapped out on it) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, note that available swap is only potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space. Consequently, 100 percent utilization on a single device is not always obtainable.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

GBL_SWAP_SPACE_UTIL_HIGH

The highest average percentage of available swap space currently in use (has memory belonging to processes paged or swapped out on it) in any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, note that available swap is only potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space. Consequently, 100 percent utilization on a single device is not always obtainable.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

GBL_SYSCALL

The number of system calls during the interval.

High system call rates are normal on busy systems, especially with IO intensive applications. Abnormally high system call rates may indicate problems such as a “hung” terminal that is stuck in a loop generating read system calls.

GBL_SYSCALL_RATE

The average number of system calls per second during the interval.

High system call rates are normal on busy systems, especially with IO intensive applications. Abnormally high system call rates may indicate problems such as a “hung” terminal that is stuck in a loop generating read system calls.

On HP-UX, system call rates affect the overhead of the midaemon.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

On HP-UX, compare this metric to GBL_DISK_LOGL_IO_RATE to see if high system callrates correspond to high disk IO. GBL_CPU_SYSCALL_UTIL shows the CPU utilization due to processing system calls.

GBL_SYSCALL_RATE_CUM

The average number of system calls per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

GBL_SYSCALL_RATE_HIGH

The highest number of system calls per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

GBL_SYSTEM_ID

The network node hostname of the system. This is the same as the output from the “uname -n” command.

On Windows, the name obtained from GetComputerName.

GBL_SYSTEM_TYPE

On Unix systems, this is either the model of the system or the instruction set architecture of the system.

On Windows, this is the processor architecture of the system.

GBL_SYSTEM_UPTIME_HOURS

The time, in hours, since the last system reboot.

GBL_SYSTEM_UPTIME_SECONDS

The time, in seconds, since the last system reboot.

GBL_SYS_QUEUE

The average number of processes or kernel threads blocked on SYSTM (that is, system resources) during the interval. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on SYSTM divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being

examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SYS_WAIT_PCT

The percentage of time processes or kernel threads were blocked on SYSTM (that is, system resources) during the interval. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on SYSTM divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_SYS_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on SYSTM (that is, system resources) during the interval. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

GBL_TERM_IO_QUEUE

The average number of processes or kernel threads blocked on terminal IO (waiting for their terminal IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on TERM (that is, terminal IO) divided by the interval time.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_TERM_IO_WAIT_PCT

The percentage of time processes or kernel threads were blocked on terminal IO (waiting for terminal IO to complete) during the interval.

This is calculated as the accumulated time that all processes or kernel threads spent blocked on TERM (that is, terminal IO) divided by the accumulated time that all processes or kernel threads were alive during the interval.

The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

No direct comparison is reasonable with the Application WAIT PCT metrics since they represent percentages within the context of a specific application and cannot be summed or compared with global values easily. In addition, the sum of each Application WAIT PCT for all applications will not equal 100% since these values will vary greatly depending on the number of processes or kernel threads in each application.

For example, the GBL_DISK_SUBSYSTEM_QUEUE values can be low, while the APP_DISK_SUBSYSTEM_WAIT_PCT values can be high. In this case, there are many processes on the system, but there are only a very small number of processes in the specific application that is being examined and there is a high percentage of those few processes that are blocked on the disk I/O subsystem.

GBL_TERM_IO_WAIT_TIME

The accumulated time, in seconds, that all processes or kernel threads were blocked on terminal IO (waiting for their terminal IO to complete) during the interval.

GBL_THRESHOLD_PROCCPU

The process CPU threshold specified in the parm file.

GBL_THRESHOLD_PROCDISK

The process disk threshold specified in the parm file.

GBL_THRESHOLD_PROCIO

The process IO threshold specified in the parm file.

GBL_THRESHOLD_PROCMEM

The process memory threshold specified in the parm file.

GBL_TT_OVERFLOW_COUNT

The number of new transactions that could not be measured because the Measurement Processing Daemon's (midaemon) Measurement Performance Database is full. If this happens, the default Measurement Performance Database size is not large enough to hold all of the registered transactions on this system. This can be remedied by stopping and restarting the midaemon process using the -smdvss option to specify a larger Measurement Performance Database size. The current Measurement Performance Database size can be checked using the midaemon -sizes option.

LDOM_ACTIVE

This metric indicates whether the Locality Domain is active or not.

LDOM_ID

The identifier for the Locality Domain. This identifier is 'na' for global and cross-LDOM memory.

LDOM_MEM_AVAIL

The amount of physical memory avail in the Locality Domain.

LDOM_MEM_AVAIL_DEL

The amount of memory that can be on-line deleted from the Locality Domain.

LDOM_MEM_FREE

The amount of free memory in the Locality Domain.

Locality Domain metrics are available on HP-UX 11iv2 and above. GBL_MEM_FREE and LDOM_MEM_FREE, as well as the memory utilization metrics derived from them, may not always fully match. GBL_MEM_FREE represents free memory in the kernel's reservation layer while LDOM_MEM_FREE shows actual free pages. If memory has been reserved but not actually consumed from the Locality Domains, the two values won't match. Because GBL_MEM_FREE includes pre-reserved memory, the GBL_MEM_* metrics are a better indicator of actual memory consumption in most situations.

LDOM_MEM_FREE_DEL

The amount of free memory that can be on-line deleted from the Locality Domain.

LDOM_MEM_TYPE

LDOM_MEM_UTIL

The percentage of memory in use in the Locality Domain during the interval

Locality Domain metrics are available on HP-UX 11iv2 and above. GBL_MEM_FREE and LDOM_MEM_FREE, as well as the memory utilization metrics derived from them, may not always fully match. GBL_MEM_FREE represents free memory in the kernel's reservation layer while LDOM_MEM_FREE shows actual free pages. If memory has been reserved but not actually consumed from the Locality Domains, the two values won't match. Because GBL_MEM_FREE includes pre-reserved memory, the GBL_MEM_* metrics are a better indicator of actual memory consumption in most situations.

LDOM_MEM_UTIL_HIGH

The highest percentage of memory in the Locality Domain in use during any interval over the cumulative collection time.

LDOM_NUM_CPU

The number of enabled CPUs in the Locality Domain

LDOM_PHYS_ID

The architecture dependent physical identifier for the Locality Domain. This identifier is 'na' for global memory.

LVDETAIL_LABEL

The type of entry this volume group or logical volume is associated with, which can be a device, partition, file system, logical volume, or volume group.

LVDETAIL_NAME

The name of the device, partition, file system, logical volume, or volume group this volume group or logical volume is associated with.

LV_AVG_READ_SERVICE_TIME

The average time, in milliseconds, that this logical volume spent processing each read request during the interval. For example, a value of 5.14 would indicate that read requests during the last interval took on average slightly longer than five one-thousandths of a second to complete for this device.

This metric can be used to help determine which logical volumes are taking more time than usual to process requests.

This metric is reported as “na” for LVM.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_AVG_WRITE_SERVICE_TIME

The average time, in milliseconds, that this logical volume spent processing each write request during the interval. For example, a value of 5.14 would indicate that write requests during the last interval took on average slightly longer than five one-thousandths of a second to complete for this device.

This metric can be used to help determine which logical volumes are taking more time than usual to process requests.

This metric is reported as “na” for LVM.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_CACHE_HIT

The number of requests successfully satisfied from the Mirror Write Cache (MWC) during the interval.

The Mirror Write Cache tracks each write of mirrored data to the physical volumes and maintains a record of any mirrored writes not yet successfully completed at the time of a system crash.

This metric is reported as “na” for VERITAS Volume Manager.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_CACHE_MISS

The number of requests that were not satisfied from the Mirror Write Cache (MWC) during the interval.

The MWC is disabled with the lvchange(1M) command (“lvchange -M n...”), which may increase system performance, but slow down recovery in the event of a system failure.

This metric is reported as “na” for VERITAS Volume Manager.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_CACHE_QUEUE

The number of requests queued to the Mirror Write Cache (MWC) at the end of the interval.

The MWC is only used for volume mirroring and its use degrades performance, as extra work is required during disk writes to maintain the Mirror Write Cache.

The MWC is disabled with the lvchange(1M) command (“lvchange -M n...”), which may increase system performance, but slow down recovery in the event of a system failure.

This metric is reported as “na” for VERITAS Volume Manager.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_CACHE_SIZE

The number of entries in this logical volume group's Mirror Write Cache (MWC). The size of this cache is determined by the kernel's logical volume code and is not configurable.

The MWC is optional and only used for volume mirroring. The MWC tracks each write of mirrored data to the physical volumes and maintains a record of any mirrored writes not yet successfully completed at the time of a system crash.

The MWC is disabled with the lvchange(1M) command (“lvchange -M n...”), which may increase system performance, but slow down recovery in the event of a system failure.

This metric is reported as “na” for VERITAS Volume Manager.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_DEVNO

Major / Minor number of this logical volume.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

Disk groups in the VERITAS Volume Manager do not have device files. Therefore, “na” is reported for this metric since it is not applicable.

LV_DIRNAME

The path name of this logical volume or volume/disk group.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

For LVM logical volumes, this is the name used as a parameter to the lvdisplay(1M) command. For volume groups, this is the name used as a parameter to the vgdisplay(1M) command.

The entry referred to as the “/dev/vgXX/group” entry shows the internal resources used by the LVM software to manage the logical volumes.

LV_GROUP_NAME

On HP-UX, this is the name of this volume/disk group associated with a logical volume.

On SUN and AIX, this is the name of this volume group associated with a logical volume. On SUN, this metric is applicable only for the Veritas LVM.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_INTERVAL

The amount of time in the interval.

LV_INTERVAL_CUM

The amount of time over the cumulative collection time, or since the last configuration change.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

LV_OPEN_LV

The number of logical volumes currently opened in this volume group (or disk group, if HP-UX). An entry of “na” indicates that there are no logical volumes open in this volume group and there are no active disks in this volume group.

On HP-UX, the extra entry (referred to as the “/dev/vgXX/group” entry), shows the internal resources used by the LVM software to manage the logical volumes.

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

On SUN, this metric is reported as “na” for logical volumes and metadevices since it is not applicable.

LV_READ_BYTE_RATE

The number of physical KBs per second read from this logical volume during the interval.

Note that bytes read from the buffer cache are not included in this calculation.

LV_READ_BYTE_RATE_CUM

The average number of physical KBs per second read from this logical volume over the cumulative collection time, or since the last configuration change.

Note that bytes read from the buffer cache are not included in this calculation.

On SUN, DiskSuite metadevices are not supported. This metric is reported as “na” for volume groups since it is not applicable.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

LV_READ_RATE

The number of physical reads per second for this logical volume during the interval.

This may not correspond to the physical read rate from a particular disk drive since a logical volume may be composed of many disk drives or it may be a subset of a disk drive. An individual physical read from one logical volume may span multiple individual disk drives.

Since this is a physical read rate, there may not be any correspondence to the logical read rate since many small reads are satisfied in the buffer cache, and large logical read requests must be broken up into physical read requests.

LV_READ_RATE_CUM

The average number of physical reads per second for this volume over the cumulative collection time, or since the last configuration change.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

LV_TYPE

Either “G” or “V”, indicating either a volume/disk group (“G”) or a logical volume (“V”). On SUN, it can also be a Disk Suite meta device (“S”).

On HP-UX 11i and beyond, data is available from VERITAS Volume Manager (VxVM). LVM (Logical Volume Manager) uses the terminology “volume group” to describe a set of related volumes. VERITAS Volume Manager uses the terminology “disk group” to describe a collection of VM disks. For additional information on VERITAS Volume Manager, see vxintro(1M).

LV_WRITE_BYTE_RATE

The number of KBs per second written to this logical volume during the interval.

LV_WRITE_BYTE_RATE_CUM

The average number of KBs per second written to this logical volume over the cumulative collection time, or since the last configuration change.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

LV_WRITE_RATE

The number of physical writes per second to this logical volume during the interval.

This may not correspond to the physical write rate to a particular disk drive since a logical volume may be composed of many disk drives or it may be a subset of a disk drive.

Since this is a physical write rate, there may not be any correspondence to the logical write rate since many small writes are combined in the buffer cache, and many large logical writes must be broken up.

LV_WRITE_RATE_CUM

The average number of physical writes per second to this volume over the cumulative collection time, or since the last configuration change.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

PRM_BYVG_GROUP_ENTITLEMENT

The PRM Disk entitlement for this PRM Group ID entry as defined in the PRM configuration file. There must be exactly one volume group record for every PRM group record. The sum of the disk entitlements must be 100 percent for each volume group.

PRM_BYVG_GROUP_UTIL

A group's current percentage of disk bandwidth relative to other PRM groups' usage of the same volume group.

PRM_BYVG_INTERVAL

The amount of time in the interval.

PRM_BYVG_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

PRM_BYVG_PRM_GROUPID

The PRM Group ID. The PRM Group ID is kept in the PRM configuration file.

PRM_BYVG_PRM_GROUPNAME

The PRM group name. The PRM group name is kept in the PRM configuration file.

PRM_BYVG_REQUEST

The number of KBs (or MBs if specified) the PRM group requested to have read from or written to the logical volumes in the current volume group during the interval.

The PRM_BYVG_* metrics report on the total bytes requested/transferred for a specified volume group. The byte counts are the total of various IO requests which result in physical IO activity. These requests may include:

```
  - Raw IO directed to a raw logical
    volume
  - Delayed Buffer Cache writes
  - Buffer Cache misses that cause
    reads
  - Large IO that bypasses buffer
    cache
  - Virtual Memory Paging Activity
```

Since the PRM configuration is dynamic, the collection may be restarted. Two intervals are required before the new values are reported. The first interval after the collection is restarted displays n/a (not available) for all of the counts.

PRM_BYVG_REQUEST_CUM

The number of KBs (or MBs if specified) the PRM group requested be read from or written to the logical volumes in the current volume group over the cumulative collection time, or since the last configuration change.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

The PRM_BYVG_* metrics report on the total bytes requested/transferred for a specified volume group. The byte counts are the total of various IO requests which result in physical IO activity. These requests may include:

```
 - Raw IO directed to a raw logical
   volume
 - Delayed Buffer Cache writes
 - Buffer Cache misses that cause
   reads
 - Large IO that bypasses buffer
   cache
 - Virtual Memory Paging Activity
```

Since the PRM configuration is dynamic, the collection may be restarted. Two intervals are required before the new values are reported. The first interval after the collection is restarted displays n/a (not available) for all of the counts.

PRM_BYVG_REQUEST_QUEUE

The request queue length for the specified volume group.

PRM_BYVG_TRANSFER

The number of KBs (or MBs if specified) the PRM group has read from or written to the logical volumes in the current volume group during the interval.

The PRM_BYVG_* metrics report on the total bytes requested/transferred for a specified volume group. The byte counts are the total of various IO requests which result in physical IO activity. These requests may include:

```
- Raw IO directed to a raw logical
  volume
- Delayed Buffer Cache writes
- Buffer Cache misses that cause
  reads
- Large IO that bypasses buffer
  cache
- Virtual Memory Paging Activity
```

Since the PRM configuration is dynamic, the collection may be restarted. Two intervals are required before the new values are reported. The first interval after the collection is restarted displays n/a (not available) for all of the counts.

PRM_BYVG_TRANSFER_CUM

The number of KBs (or MBs if specified) the PRM group has read from or written to the logical volumes in the current volume group over the cumulative collection time, or since the last configuration change.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

The PRM_BYVG_* metrics report on the total bytes requested/transferred for a specified volume group. The byte counts are the total of various IO requests which result in physical IO activity. These requests may include:

```
- Raw IO directed to a raw logical
  volume
- Delayed Buffer Cache writes
- Buffer Cache misses that cause
  reads
```

```
 - Large IO that bypasses buffer
   cache
 - Virtual Memory Paging Activity
```

Since the PRM configuration is dynamic, the collection may be restarted. Two intervals are required before the new values are reported. The first interval after the collection is restarted displays n/a (not available) for all of the counts.

PROCSYSCALL_ACTIVE_CUM

The number of different system calls called by this process during the time it has been enabled for system call profiling.

PROCSYSCALL_CALL_COUNT

The number of system calls made to this function by this process during the interval.

PROCSYSCALL_CALL_COUNT_CUM

The number of system calls made by this process to this function during the time it has been enabled for system call profiling.

PROCSYSCALL_CALL_ID

The ID number of the system call. System calls are sequentially numbered starting with one.

PROCSYSCALL_CALL_NAME

The system call name.

PROCSYSCALL_CALL_RATE

The number of system calls per second made by this process to this function during the last interval.

PROCSYSCALL_CALL_RATE_CUM

The average number of system calls per second made by this process to this function during the time it has been enabled for system call profiling.

PROCSYSCALL_INTERVAL

The amount of time in the interval.

PROCSYSCALL_INTERVAL_CUM

The time, in seconds, system call data has been collected for this process.

PROCSYSCALL_TOTAL_TIME

The elapsed time, in seconds, this process was in this system call. This value maybe greater then the interval time since the system call may have been started before the interval started.

PROCSYSCALL_TOTAL_TIME_CUM

The total elapsed time, in seconds, that this process was in this system call. This value maybe greater than the cumulative interval time since the system call may have been started before data collection.

PROC_APP_ID
THREAD_APP_ID

The ID number of the application to which the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) belonged during the interval.

Application “other” always has an ID of 1. There can be up to 999 user-defined applications, which are defined in the parm file.

PROC_APP_NAME
THREAD_APP_NAME

The application name of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

Processes (or kernel threads, if HP-UX/Linux Kernel 2.6 and above) are assigned into application groups based upon rules in the parm file. If a process does not fit any rules in this file, it is assigned to the application “other.”

The rules include decisions based upon pathname, user ID, priority, and so forth. As these values change during the life of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above), it is re-assigned to another application. This re-evaluation is done every measurement interval.

PROC_CACHE_WAIT_PCT
THREAD_CACHE_WAIT_PCT

The percentage of time the process or kernel thread was blocked on CACHE (waiting for the file systembuffer cache to be updated) during the interval. Processes or kernel threads doing raw IO to a disk are not included in this measurement. Processes and kernel threads doing buffered IO to disks normally spend more time blocked on CACHE and IO than on DISK.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_CACHE_WAIT_PCT_CUM
THREAD_CACHE_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on CACHE (waiting for the file systembuffer cache to be updated) over the cumulative collection time. Processes or kernel threads doing raw IO to a disk are not included in this measurement.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_CACHE_WAIT_TIME
THREAD_CACHE_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on CACHE (waiting for the file systembuffer cache to be updated) during the interval. Processes or kernel threads doing raw IO to a disk are not included in this measurement.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_CACHE_WAIT_TIME_CUM
THREAD_CACHE_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on CACHE (waiting for the file systembuffer cache to be updated) over the cumulative collection time. Processes or kernel threads doing raw IO to a disk are not included in this measurement.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_CDFS_WAIT_PCT
THREAD_CDFS_WAIT_PCT

The percentage of time the process or kernel thread was blocked on CDFS (waiting for its Compact Disk file system IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_CDFS_WAIT_PCT_CUM
THREAD_CDFS_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on CDFS (waiting for its Compact Disk file system IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_CDFS_WAIT_TIME
THREAD_CDFS_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on CDFS (waiting in the CD-ROM driver for Compact Disc file system IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_CDFS_WAIT_TIME_CUM
THREAD_CDFS_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on CDFS (waiting in the CD-ROM driver for Compact Disc file system IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_CLOSE
THREAD_CLOSE

The number of file closes made by the process or kernel thread during the interval. This corresponds to the number of close(2) system calls.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_CLOSE_CUM
THREAD_CLOSE_CUM

The number of file closes made by the process or kernel thread over the cumulative collection time. This corresponds to the number of close(2) system calls.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_CPU_ALIVE_SYS_MODE_UTIL
THREAD_CPU_ALIVE_SYS_MODE_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) in system mode as a percentage of the time it is alive during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_ALIVE_TOTAL_UTIL THREAD_CPU_ALIVE_TOTAL_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) as a percentage of the time it is alive during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_ALIVE_USER_MODE_UTIL THREAD_CPU_ALIVE_USER_MODE_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) in user mode as a percentage of the time it is alive during the interval. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_CSWITCH_TIME
THREAD_CPU_CSWITCH_TIME

The time, in seconds, that the process or kernel thread spent in context switching during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_CSWITCH_TIME_CUM
THREAD_CPU_CSWITCH_TIME_CUM

The time, in seconds, that the selected process or kernel thread spent in context switching over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_CSWITCH_UTIL
THREAD_CPU_CSWITCH_UTIL

The percentage of time spent in context switching the current process or kernel thread during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_CSWITCH_UTIL_CUM
THREAD_CPU_CSWITCH_UTIL_CUM

The average percentage of time spent in context switching the process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_INTERRUPT_TIME THREAD_CPU_INTERRUPT_TIME

The time, in seconds, that the process or kernel thread spent processing interrupts during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_INTERRUPT_TIME_CUM THREAD_CPU_INTERRUPT_TIME_CUM

The time, in seconds, that the process or kernel thread spent processing interrupts over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_INTERRUPT_UTIL
THREAD_CPU_INTERRUPT_UTIL

The percentage of time that this process or kernel thread was in interrupt mode during the last interval. Interrupt mode means that interrupts were being handled while the process or kernel thread was loaded and running on the CPU. The interrupts may have been generated by any process, not just the running process, but they were handled while the process or kernel thread was running and may have had an impact on the performance of this process or kernel thread.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be

added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_INTERRUPT_UTIL_CUM THREAD_CPU_INTERRUPT_UTIL_CUM

The average percentage of time that this process or kernel thread was in interrupt mode over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_LAST_USED THREAD_CPU_LAST_USED

The ID number of the processor that last ran the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above). For uni-processor systems, this value is always zero.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

PROC_CPU_NICE_TIME THREAD_CPU_NICE_TIME

The time, in seconds, that this niced process or kernel thread was using the CPU in user mode during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NICE_TIME_CUM
THREAD_CPU_NICE_TIME_CUM

The time, in seconds, that this niced process or kernel thread was in user mode over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NICE_UTIL
THREAD_CPU_NICE_UTIL

The percentage of time that this niced process or kernel thread was in user mode during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NICE_UTIL_CUM
THREAD_CPU_NICE_UTIL_CUM

The average percentage of time that this niced process or kernel thread was in user mode over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NNICE_TIME
THREAD_CPU_NNICE_TIME

The time, in seconds, that this negatively niced process or kernel thread was using the CPU in user mode during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NNICE_TIME_CUM
THREAD_CPU_NNICE_TIME_CUM

The time, in seconds, that this negatively niced process or kernel thread was in user mode over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NNICE_UTIL
THREAD_CPU_NNICE_UTIL

The percentage of time that this negatively niced process or kernel thread was in user mode during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NNICE_UTIL_CUM THREAD_CPU_NNICE_UTIL_CUM

The average percentage of time that this negatively niced process or kernel thread was in user mode over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NORMAL_TIME
THREAD_CPU_NORMAL_TIME

The time, in seconds, that the selected process or kernel thread was in user mode at normal priority during the interval. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NORMAL_TIME_CUM
THREAD_CPU_NORMAL_TIME_CUM

The time, in seconds, that the selected process or kernel thread was in user mode at normal priority over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NORMAL_UTIL
THREAD_CPU_NORMAL_UTIL

The percentage of time that this process or kernel thread was in user mode at a normal priority during the interval. “At a normal priority” means the neither rtprio or nice had been used to alter the priority of the process or kernel thread during the interval. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_NORMAL_UTIL_CUM
THREAD_CPU_NORMAL_UTIL_CUM

The average percentage of time a process or kernel thread was in user mode at normal priority over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be

added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_REALTIME_TIME THREAD_CPU_REALTIME_TIME

The time, in seconds, that the selected process or kernel thread was in user mode at a realtime priority during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_REALTIME_TIME_CUM THREAD_CPU_REALTIME_TIME_CUM

The time, in seconds, that the selected process or kernel thread was in user mode at a realtime priority over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_REALTIME_UTIL
THREAD_CPU_REALTIME_UTIL

The percentage of time that this process or kernel thread was at a realtime priority during the interval. The realtime CPU is separated out to allow users to see the effect of using the realtime facilities to alter priority.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_REALTIME_UTIL_CUM
THREAD_CPU_REALTIME_UTIL_CUM

The percentage of time that the CPU was in user mode executing the current process or kernel thread at a realtime priority over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SWITCHES
THREAD_CPU_SWITCHES

The number of times the process or kernel thread was switched to another processor during the interval. For uni-processor systems, this value is always zero.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_CPU_SWITCHES_CUM
THREAD_CPU_SWITCHES_CUM

The number of times the process or kernel thread was switched to another processor over the cumulative collection time. For uni-processor systems, this value is always zero.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_CPU_SYSCALL_TIME
THREAD_CPU_SYSCALL_TIME

The time, in seconds, that this process or kernel thread spent executing system calls in system mode, excluding interrupt or context processing, during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYSCALL_TIME_CUM
THREAD_CPU_SYSCALL_TIME_CUM

The time, in seconds, that this process or kernel thread spent executing system calls in system mode, excluding interrupt or context processing, over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYSCALL_UTIL
THREAD_CPU_SYSCALL_UTIL

The percentage of the total CPU time this process or kernel thread spent in system mode (excluding interrupt, context switch, trap, or vfault CPU) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYSCALL_UTIL_CUM THREAD_CPU_SYSCALL_UTIL_CUM

The average percentage of the total CPU time this process or kernel thread spent in system mode (excluding interrupt, context switch, trap, or vfault CPU) during the interval.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYS_MODE_TIME THREAD_CPU_SYS_MODE_TIME

The CPU time in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYS_MODE_TIME_CUM THREAD_CPU_SYS_MODE_TIME_CUM

The CPU time in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYS_MODE_UTIL
THREAD_CPU_SYS_MODE_UTIL

The percentage of time that the CPU was in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

High system mode CPU utilizations are normal for IO intensive programs. Abnormally high system CPU utilization can indicate that a hardware problem is causing a high interrupt rate. It can also indicate programs that are not using system calls efficiently.

A classic “hung shell” shows up with very high system mode CPU because it gets stuck in a loop doing terminal reads (a system call) to a device that never responds.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYS_MODE_UTIL_CUM
THREAD_CPU_SYS_MODE_UTIL_CUM

The average percentage of time that the CPU was in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_TIME
THREAD_CPU_TOTAL_TIME

The total CPU time, in seconds, consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On HP-UX, the total CPU time is the sum of the CPU time components for a process or kernel thread, including system, user, context switch, interrupts processing, realtime, and nice utilization values.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_TIME_CUM THREAD_CPU_TOTAL_TIME_CUM

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time. CPU time is in seconds unless otherwise specified.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This is calculated as

```
PROC_CPU_TOTAL_TIME_CUM =
  PROC_CPU_SYS_MODE_TIME_CUM +
  PROC_CPU_USER_MODE_TIME_CUM
```

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_UTIL
THREAD_CPU_TOTAL_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) as a percentage of the total CPU time available during the interval.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On HP-UX, the total CPU utilization is the sum of the CPU utilization components for a process or kernel thread, including system, user, context switch, interrupts processing, realtime, and nice utilization values.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_UTIL_CUM
THREAD_CPU_TOTAL_UTIL_CUM

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) as a percentage of the total CPU time available over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On HP-UX, the total CPU utilization is the sum of the CPU utilization components for a process or kernel thread, including system, user, context switch, interrupts processing, realtime, and nice utilization values.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TRAP_COUNT THREAD_CPU_TRAP_COUNT

The number of times the CPU was in trap handler code for this process or kernel thread during the interval.

On HP-UX, all exceptions (including faults) cause traps. These include pfaults (protection faults), vfaults (virtual faults), time slice expiration (rescheduling), zero divide, illegal or privileged instructions, single-stepping, breakpoints, and so on. The kernel trap handler code will switch trap counters for vfaults and pfaults to fault counters when appropriate. As such, the trap count excludes vfaults and pfaults.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_CPU_TRAP_COUNT_CUM THREAD_CPU_TRAP_COUNT_CUM

The number of times the CPU was in trap handler code for this process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, all exceptions (including faults) cause traps. These include pfaults (protection faults), vfaults (virtual faults), time slice expiration (rescheduling), zero divide, illegal or privileged instructions, single-stepping, breakpoints, and so on. The kernel trap handler code will switch trap counters for vfaults and pfaults to fault counters when appropriate. As such, the trap count excludes vfaults and pfaults.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_CPU_USER_MODE_TIME
THREAD_CPU_USER_MODE_TIME

The time, in seconds, the process (or kernel threads, if HP-UX/Linux Kernel 2.6 and above) was using the CPU in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_USER_MODE_TIME_CUM
THREAD_CPU_USER_MODE_TIME_CUM

The time, in seconds, the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was using the CPU in user mode over the cumulative collection time. collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_USER_MODE_UTIL THREAD_CPU_USER_MODE_UTIL

The percentage of time the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was using the CPU in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_USER_MODE_UTIL_CUM THREAD_CPU_USER_MODE_UTIL_CUM

The average percentage of time the process (or kernel thread, if HP_UX/Linux Kernel 2.6 and above) was using the CPU in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_DISK_FS_READ
THREAD_DISK_FS_READ

Number of file system physical disk reads made by a process or kernel thread during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical reads generated by user file system access and do not include virtual memory reads, system reads (inode access), or reads relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical reads in this category. They appear under virtual memory reads.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_FS_READ_CUM THREAD_DISK_FS_READ_CUM

Number of file system physical disk reads made by a process or kernel thread over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are physical reads generated by user file system access and do not include virtual memory reads, system reads (inode access), or reads relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical reads in this category. They appear under virtual memory reads.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_FS_READ_RATE THREAD_DISK_FS_READ_RATE

The number of file system physical disk reads made by a process or kernel thread during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical reads generated by user file system access and do not include virtual memory reads, system reads (inode access), or reads relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical reads in this category. They appear under virtual memory reads.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_FS_WRITE
THREAD_DISK_FS_WRITE

Number of file system physical disk writes made by a process or kernel thread during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical writes generated by user file system access and do not include virtual memory writes, system writes (inode updates), or writes relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical writes in this category. They appear under virtual memory writes.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_FS_WRITE_CUM
THREAD_DISK_FS_WRITE_CUM

Number of file system physical disk writes made by a process or kernel thread over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are physical writes generated by user file system access and do not include virtual memory writes, system writes (inode updates), or writes relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical writes in this category. They appear under virtual memory writes.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_FS_WRITE_RATE
THREAD_DISK_FS_WRITE_RATE

The number of file system physical disk writes made by a process or kernel thread during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical writes generated by user file system access and do not include virtual memory writes, system writes (inode updates), or writes relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical writes in this category. They appear under virtual memory writes.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_IO
THREAD_DISK_LOGL_IO

The number of logical IOs made by (or for) a process or kernel thread during the interval. NFS mounted disks are not included in this list.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk).

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_IO_CUM
THREAD_DISK_LOGL_IO_CUM

The number of logical IOs made by (or for) a process or kernel thread over the cumulative collection time. NFS mounted disks are not included in this list.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk).

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_IO_RATE
THREAD_DISK_LOGL_IO_RATE

The number of logical IOs per second made by (or for) a process or kernel thread during the interval. NFS mounted disks are not included in this list.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

For processes which run for less than the measurement interval, this metric is normalized over the measurement interval. For example, a process ran for 1 second and did 50 IOs during its life. If the measurement interval is 5 seconds, it is reported as having done 10 IOs per second. If the measurement interval is 60 seconds, it is reported as having done 50/60 or 0.83 IOs per second.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk).

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_IO_RATE_CUM
THREAD_DISK_LOGL_IO_RATE_CUM

The average number of logical IOs per second made by (or for) a process or kernel thread over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

On many Unix systems, there are several reasons why logical IOs may not correspond with physical IOs. Logical IOs may not always result in a physical disk access, since the data may already reside in memory -- either in the buffer cache, or in virtual memory if the IO is to a memory mapped file. Several logical IOs may all map to the same physical page or block. In these two cases, logical IOs are greater than physical IOs.

The reverse can also happen. A single logical write can cause a physical read to fetch the block to be updated from disk, and then cause a physical write to put it back on disk. A single logical IO can require more than one physical page or block, and these can be found on different disks. Mirrored disks further distort the relationship between logical and physical IO, since physical writes are doubled.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_READ
THREAD_DISK_LOGL_READ

The number of disk logical reads made by a process or kernel thread during the interval. Calls destined for NFS mounted files are not counted.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_READ_CUM
THREAD_DISK_LOGL_READ_CUM

The number of disk logical reads made by a process or kernel thread over the cumulative collection time. Calls destined for NFS mounted files are not counted.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_READ_RATE
THREAD_DISK_LOGL_READ_RATE

The number of logical reads per second made by (or for) a process or kernel thread during the interval. Calls destined for NFS mounted files are not counted.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk).

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_WRITE
THREAD_DISK_LOGL_WRITE

Number of disk logical writes made by a process or kernel thread during the interval. Calls destined for NFS mounted files are not counted.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_WRITE_CUM
THREAD_DISK_LOGL_WRITE_CUM

Number of disk logical writes made by a process or kernel thread over the cumulative collection time. Calls destined for NFS mounted files are not counted.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_LOGL_WRITE_RATE THREAD_DISK_LOGL_WRITE_RATE

The number of logical writes per second made by (or for) a process or kernel thread during the interval. NFS mounted disks are not included in this list.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk).

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_PHYS_IO_RATE THREAD_DISK_PHYS_IO_RATE

The average number of physical disk IOs per second made by the process or kernel thread during the interval.

For processes which run for less than the measurement interval, this metric is normalized over the measurement interval. For example, a process ran for 1 second and did 50 IOs during its life. If the measurement interval is 5 seconds, it is reported as having done 10 IOs per second. If the measurement interval is 60 seconds, it is reported as having done 50/60 or 0.83 IOs per second.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_IO_RATE_CUM
THREAD_DISK_PHYS_IO_RATE_CUM

The number of physical disk IOs per second made by the selected process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_READ
THREAD_DISK_PHYS_READ

The number of physical reads made by (or for) a process or kernel thread during the last interval.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_READ_CUM
THREAD_DISK_PHYS_READ_CUM

The number of physical reads made by (or for) a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_READ_RATE
THREAD_DISK_PHYS_READ_RATE

The number of physical reads per second made by (or for) a process or kernel thread during the interval.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_WRITE
THREAD_DISK_PHYS_WRITE

The number of physical writes made by (or for) a process or kernel thread during the last interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_WRITE_CUM
THREAD_DISK_PHYS_WRITE_CUM

The number of physical writes made by (or for) a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_WRITE_RATE
THREAD_DISK_PHYS_WRITE_RATE

The number of physical writes per second made by (or for) a process or kernel thread during the interval.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_RAW_READ
THREAD_DISK_RAW_READ

Number of raw reads made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_RAW_READ_CUM
THREAD_DISK_RAW_READ_CUM

Number of raw reads made for a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_RAW_READ_RATE THREAD_DISK_RAW_READ_RATE

Rate of raw reads made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_RAW_WRITE THREAD_DISK_RAW_WRITE

Number of raw writes made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_RAW_WRITE_CUM THREAD_DISK_RAW_WRITE_CUM

Number of raw writes made for a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_RAW_WRITE_RATE THREAD_DISK_RAW_WRITE_RATE

Rate of raw writes made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_LOGL_READ THREAD_DISK_REM_LOGL_READ

The number of remote logical reads made by a process or kernel thread during the last interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_LOGL_READ_CUM
THREAD_DISK_REM_LOGL_READ_CUM

The number of remote logical reads made by a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_LOGL_READ_RATE
THREAD_DISK_REM_LOGL_READ_RATE

The number of remote logical reads per second made by (or for) a process or kernel thread during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_LOGL_WRITE
THREAD_DISK_REM_LOGL_WRITE

Number of remote logical writes made by a process or kernel thread during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_LOGL_WRITE_CUM
THREAD_DISK_REM_LOGL_WRITE_CUM

Number of remote logical writes made by a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_LOGL_WRITE_RATE THREAD_DISK_REM_LOGL_WRITE_RATE

The number of remote logical writes per second made by (or for) a process or kernel thread during the interval.

On HP-UX, the remote logical IOs include all IO requests generated on a local client to a remotely mounted file system or disk. If the logical request is satisfied on the local client (that is, the data is in a local memory buffer), a physical request is not generated. Otherwise, a physical IO request is made to the remote machine to read/write the data. Note that, in either case, a logical IO request is made.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_PHYS_READ THREAD_DISK_REM_PHYS_READ

The number of remote physical reads made by (or for) a process or kernel thread during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_PHYS_READ_CUM THREAD_DISK_REM_PHYS_READ_CUM

The number of remote physical reads made by (or for) a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_PHYS_READ_RATE THREAD_DISK_REM_PHYS_READ_RATE

The number of remote physical reads per second made by (or for) a process or kernel thread during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_PHYS_WRITE THREAD_DISK_REM_PHYS_WRITE

The number of physical writes made by (or for) a process or kernel thread during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_PHYS_WRITE_CUM THREAD_DISK_REM_PHYS_WRITE_CUM

The number of physical writes made by (or for) a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_REM_PHYS_WRITE_RATE
THREAD_DISK_REM_PHYS_WRITE_RATE

The number of physical writes per second made by (or for) a process or kernel thread during the interval.

On HP-UX, if an IO cannot be satisfied in a local client machine's memory buffer, a remote physical IO request is generated. This may or may not require a physical disk IO on the remote system. In either case, the remote IO request is considered a physical request on the local client machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_SUBSYSTEM_WAIT_PCT
THREAD_DISK_SUBSYSTEM_WAIT_PCT

The percentage of time the process or kernel thread was blocked on the disk subsystem (waiting for its file system IOs to complete) during the interval. This includes time spent waiting in the DISK, INODE, CACHE, and CDFS wait states. It does not include processes doing raw IO to disk devices.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_DISK_SUBSYSTEM_WAIT_PCT_CUM THREAD_DISK_SUBSYSTEM_WAIT_PCT_CUM

The percentage of time the process or kernel thread was blocked on the disk subsystem (waiting for its file system IOs to complete) over the cumulative collection time. This includes time spent waiting in the DISK, INODE, CACHE, and CDFS wait states. It does not include processes doing raw IO to disk devices.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_DISK_SUBSYSTEM_WAIT_TIME
THREAD_DISK_SUBSYSTEM_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on the disk subsystem (waiting for its file system IOs to complete) during the interval. This includes time spent waiting in the DISK, INODE, CACHE, and CDFS wait states. It does not include processes doing raw IO to disk devices.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_DISK_SUBSYSTEM_WAIT_TIME_CUM
THREAD_DISK_SUBSYSTEM_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on the disk subsystem (waiting for its file system IOs to complete) over the cumulative collection time. This includes time spent waiting in the DISK, INODE, CACHE, and CDFS wait states. It does not include processes doing raw IO to disk devices.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_DISK_SYSTEM_IO
THREAD_DISK_SYSTEM_IO

Number of file system management physical disk IOs made for a process or kernel thread during the interval.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

"Disk" in this instance refers to any locally attached physical disk drives (that is, "spindles") that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_SYSTEM_IO_RATE
THREAD_DISK_SYSTEM_IO_RATE

The number of file system management physical disk IOs per second made for a process or kernel thread during the interval.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

"Disk" in this instance refers to any locally attached physical disk drives (that is, "spindles") that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_SYSTEM_READ
THREAD_DISK_SYSTEM_READ

Number of file system management physical disk reads made for a process or kernel thread during the interval.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_SYSTEM_READ_CUM
THREAD_DISK_SYSTEM_READ_CUM

Number of file system management physical disk reads made for a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_SYSTEM_WRITE
THREAD_DISK_SYSTEM_WRITE

Number of file system management physical disk writes made for a process or kernel thread during the interval.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_SYSTEM_WRITE_CUM
THREAD_DISK_SYSTEM_WRITE_CUM

Number of file system management physical disk writes made for a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_VM_IO
THREAD_DISK_VM_IO

The number of virtual memory IOs made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_VM_IO_RATE
THREAD_DISK_VM_IO_RATE

The number of virtual memory IOs per second made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_VM_READ
THREAD_DISK_VM_READ

Number of virtual memory reads made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_VM_READ_CUM
THREAD_DISK_VM_READ_CUM

Number of virtual memory reads made for a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_VM_WRITE
THREAD_DISK_VM_WRITE

Number of virtual memory writes made for a process or kernel thread during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_VM_WRITE_CUM
THREAD_DISK_VM_WRITE_CUM

Number of virtual memory writes made for a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISK_WAIT_PCT
THREAD_DISK_WAIT_PCT

The percentage of time the process or kernel thread was blocked on DISK (waiting in the disk drivers for file system disk IO to complete) during the interval. The time spent waiting in the disk drivers is usually very small. Most of the time, processes doing file system IO are waiting on IO or CACHE. Processes waiting for character (raw) IO to a disk device are usually waiting on IO.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_DISK_WAIT_PCT_CUM THREAD_DISK_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on DISK (waiting in the disk drivers for file system disk IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_DISK_WAIT_TIME
THREAD_DISK_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on DISK (waiting in a disk driver for its disk IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_DISK_WAIT_TIME_CUM
THREAD_DISK_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on DISK (waiting in a disk driver for its disk IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_DISPATCH
THREAD_DISPATCH

The number of times the process or kernel thread was made the executing process on the CPU over the interval. This includes dispatches associated with a context switch because some other process or kernel thread had the CPU, as well as those dispatches caused by the process or kernel thread stopping, then resuming, with no other process or kernel thread running in the meantime.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_DISPATCH_CUM
THREAD_DISPATCH_CUM

The number of times the process or kernel thread was made the executing process on the CPU over the cumulative collection time. This includes dispatches associated with a context switch because some other process or kernel thread had the CPU, as well as those dispatches caused by the process or kernel thread stopping, then resuming, with no other process or kernel thread running in the meantime.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_EUID
THREAD_EUID

The Effective User ID of a process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_FILE_COUNT

The number of times this file is opened currently. Terminal devices are often opened more than once by several different processes.

PROC_FILE_MODE

A text string summarizing the type of open mode:

```
rd/wr  Opened for input & output
read   Opened for input only
write  Opened for output only
```

PROC_FILE_NAME

The path name or identifying information about the open file descriptor. If the path name string exceeds 40 characters in length, the beginning and the end of the path is shown and the middle of the name is replaced by "...".

An attempt is made to obtain the file path name by either searching the current cylinder group to find directory entries that point to the currently opened inode, or by searching the kernel name cache. Since looking up file path names would require high disk overhead, some names may not be resolved. If the path name can not be resolved, a string is returned indicating the type and inode number of the file.

For HP-UX 11.0 releases, the path name information of a file descriptor may correspond to streams device files, such as /dev/tcp and /dev/udp, which are not explicitly opened by the process. These files are opened by networking functions called by the process to access remote systems.

For the string format including an inode number, you may use the ncheck(1M) program to display the file path name relative to the mount point. Sometimes files may be deleted before they are closed. In these cases, the process file table may still have the inode even though the file is not actually present and as a result, ncheck will fail.

In the following example, note that the file system name has been included to avoid the overhead of ncheck searching all of the file systems for the inode number.

If the following file information was displayed:

Note that the following examples would all appear on one line.

```
<reg,vxfs,/var,/dev/vg00/lvol8,inode:702>
```

and then from that display, the following ncheck command was entered:

```
ncheck -i 702 -F vxfs /dev/vg00/lvol8
```

An output like the following would be generated:

```
  /dev/vg00/lvol8:
  702     /adm/cron/log
```

Since in this example /var is mounted on lvol8 of vg00, the full path name would be /var/adm/cron/log.

The string shown representing inode information when the path is not available is as follows:

```
<type,domain,filesys,volume,inode:n>
```

where:

```
The file type can be one of:
     blk      - Block device
     chr      - Character device
     dir      - Directory file
     fifo     - FIFO
     lnk      - Soft file link
     reg      - Regular file
     sock     - Socket
     emptydir - Mknod created
                directory - no
                files
     ukn      - Unknown vtype

The file domain can be one of:
     ufs      - Unix file system
     nfs      - NFS
```

```
vxfs      - Veritas file system
cdfs      - CDROM file system
nfs_spec - NFS special device
           file
nfs_bdev - NFS device file -
           block mode access
nfs_fifo - FIFO file access
           over NFS
dev_vn    - Generic vnode -
           temp type used by
           kernel
dummy     - OSF's file on file
           mount file system
pipe      - Pipe
ukn       - Unknown vfs type
```

The filesys is the file system mount point.

The volume field indicates the logical volume, if applicable.

For HP-UX 10.30 and earlier releases, if the file descriptor represents an open socket, the output format will contain the domain and protocol, followed by the IP address.

For HP-UX 11.0 and later, if the file descriptor represents a Unix address family socket which is used for IPC on the local host, its path name will be resolved if possible. For example:

```
  unix /tmp/.AgentSockets/A
```

If the local socket pathname cannot be resolved, the socket address will be shown, for example:

```
  unix -> 0x0339a200
```

For HP-UX 11.0 and later, if the file descriptor is a socket for internetwork communications (for example, udp or tcp), the socket address, domain, and protocol will be displayed respectively. The bound IP address and port number will also be displayed when available. For example:

```
<socket: 0x03189800,inet,tcp,INADDR_ANY:2121>
<socket: 0x031bd400,inet,udp,15.8.157.15:123>
```

PROC_FILE_NUMBER

The file number of the current open file.

PROC_FILE_OFFSET

The decimal value of the next access position of the current file at the end of the interval. If the open file is a tty, this is the total number of bytes sent and received since the file was first opened.

PROC_FILE_OPEN

Number of files the current process has remaining open as of the end of the interval.

PROC_FILE_TYPE

A text string describing the type of the current file. This is one of:

```
block   Block special device
chr     Character device
dir     Directory
fifo    FIFO
file    Simple file
link    Symbolic File link
network Network channel device
other   An unknown file type
pipe    Named pipe (FIFO)
reg     Regular file
socket  Socket
streams Streams
```

PROC_FORCED_CSWITCH
THREAD_FORCED_CSWITCH

The number of times that the process (or kernel thread, if HP-UX) was preempted by an external event and another process (or kernel thread, if HP-UX) was allowed to execute during the interval.

Examples of reasons for a forced switch include expiration of a time slice or returning from a system call with a higher priority process (or kernel thread, if HP-UX) ready to run.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_FORCED_CSWITCH_CUM
THREAD_FORCED_CSWITCH_CUM

The number of times the process (or kernel thread, if HP-UX) was preempted by an external event and another process (or kernel thread, if HP-UX) was allowed to execute over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Examples of reasons for a forced switch include expiration of a time slice or returning from a system call with a higher priority process (or kernel thread, if HP-UX) ready to run.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_FORK
THREAD_FORK

The total number of fork and vforksystem calls executed by this process during the interval.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_FORK_CUM
THREAD_FORK_CUM

The number of fork or vforksystem calls made by a process over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_GRAPHICS_WAIT_PCT THREAD_GRAPHICS_WAIT_PCT

The percentage of time the process or kernel thread was blocked on graphics (waiting for graphics operations to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_GRAPHICS_WAIT_PCT_CUM THREAD_GRAPHICS_WAIT_PCT_CUM

The percentage of time the process or kernel thread was blocked on graphics (waiting for graphics operations to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_GRAPHICS_WAIT_TIME
THREAD_GRAPHICS_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on graphics (waiting for their graphics operations to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_GRAPHICS_WAIT_TIME_CUM
THREAD_GRAPHICS_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on graphics (waiting for their graphics operations to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_GROUP_ID
THREAD_GROUP_ID

On most systems, this is the real group ID number of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above). On AIX, this is the effective group ID number of the process.

On HP-UX, this is the effective group ID number of the process if not in setgid mode.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_GROUP_NAME
THREAD_GROUP_NAME

The group name (from /etc/group) of a process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

The group identifier is obtained from searching the /etc/passwd file using the user ID (uid) as a key. Therefore, if more than one account is listed in /etc/passwd with the same user ID (uid) field, the first one is used. If no entry can be found for the user ID in /etc/passwd, the group name is the uid number. If no matching entry in /etc/group can be found, the group ID is returned as the group name.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_INODE_WAIT_PCT THREAD_INODE_WAIT_PCT

The percentage of time the process or kernel thread was blocked onINODE (waiting for an inode to be updated or to become available) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_INODE_WAIT_PCT_CUM THREAD_INODE_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked onINODE (waiting for an inode to be updated or to become available) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_INODE_WAIT_TIME
THREAD_INODE_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked onINODE (waiting for an inode to be updated or to become available) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_INODE_WAIT_TIME_CUM
THREAD_INODE_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked onINODE (waiting for an inode to be updated or to become available) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_INTEREST
THREAD_INTEREST

A string containing the reason(s) why the process or thread is of interest, based on the thresholds specified in the parm file.

An 'A' indicates that the process or thread exceeds the process CPU threshold, computed using the actual time the process or thread was alive during the interval.

A 'C' indicates that the process or thread exceeds the process CPU threshold, computed using the collection interval. Currently, the same CPU threshold is used for both CPU interest reasons.

A 'D' indicates that the process or thread exceeds the process disk IO threshold.

An 'I' indicates that the process or thread exceeds the IO threshold.

An 'M' indicates that the process exceeds the process memory threshold. This interest reason is only meaningful for processes and therefore not shown for threads.

New processes or threads are identified with an 'N', terminated processes or threads are identified with a 'K'.

Note that the parm file 'nonew', 'nokill' and 'shortlived' settings are logging only options and therefore ignored in Glance components.

PROC_INTERRUPTS THREAD_INTERRUPTS

The number of interrupts during the interval.

PROC_INTERRUPTS_CUM THREAD_INTERRUPTS_CUM

The number of interrupts over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

PROC_INTERVAL THREAD_INTERVAL

The amount of time in the interval. This is the same value for all processes (and kernel threads, if HP-UX/Linux Kernel 2.6 and above), regardless of whether they were alive for the entire interval.

Note, calculations such as utilizations or rates are calculated using this standardized process interval (PROC_INTERVAL), rather than the actual alive time during the interval (PROC_INTERVAL_ALIVE). Thus, if a process was only alive for 1 second and used the CPU during its entire life (1 second), but the process sample interval was 5 seconds, it would be reported as using 1/5 or 20% CPU utilization, rather than 100% CPU utilization.

PROC_INTERVAL_ALIVE THREAD_INTERVAL_ALIVE

The number of seconds that the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was alive during the interval. This may be less than the time of the interval if the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was new or died during the interval.

PROC_INTERVAL_CUM
THREAD_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On SUN, AIX, and OSF1, this differs from PROC_RUN_TIME in that PROC_RUN_TIME may not include all of the first and last sample interval times and PROC_INTERVAL_CUM does.

PROC_IOCTL
THREAD_IOCTL

The number of file ioctls made by the process during the interval. ioctls that result in data read from or written to a device are not counted. These are counted under disk and non-disk read and writes.

This metric is no longer collected on HP-UX 11.0 and beyond.

PROC_IOCTL_CUM
THREAD_IOCTL_CUM

The number of file ioctls made by the process over the cumulative collection time. ioctls that result in data read from or written to a device are not counted. These are counted under disk and non-disk reads and writes.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

This metric is no longer collected on HP-UX 11.0 and beyond.

PROC_IO_BYTE
THREAD_IO_BYTE

On HP-UX, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process or kernel thread, either directly or indirectly, during the interval.

On all other systems, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process during the interval. IOs include disk, terminal, tape and network IO.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On SUN, counts in the MB ranges in general can be attributed to disk accesses and counts in the KB ranges can be attributed to terminal IO. This is useful when looking for processes with heavy disk IO activity. This may vary depending on the sample interval length.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_IO_BYTE_CUM
THREAD_IO_BYTE_CUM

On HP-UX, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process or kernel thread, either directly or indirectly, over the cumulative collection time.

On all other systems, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process over the cumulative collection time. IOs include disk, terminal, tape and network IO.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_IO_BYTE_RATE
THREAD_IO_BYTE_RATE

On HP-UX, this is the number of physical IO KBs per second that was used by this process or kernel thread, either directly or indirectly, during the interval.

On all other systems, this is the number of physical IO KBs per second that was used by this process during the interval. IOs include disk, terminal, tape and network IO.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On SUN, counts in the MB ranges in general can be attributed to disk accesses and counts in the KB ranges can be attributed to terminal IO. This is useful when looking for processes with heavy disk IO activity. This may vary depending on the sample interval length.

Certain types of disk IOs are not counted by AIX at the process level, so they are excluded from this metric.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_IO_BYTE_RATE_CUM
THREAD_IO_BYTE_RATE_CUM

On HP-UX, this is the average number of physical IO KBs per second that was used by this process or kernel thread, either directly or indirectly, over the cumulative collection time.

On all other systems, this is the average number of physical IO KBs per second that was used by this process over the cumulative collection time. IOs include disk, terminal, tape and network IO.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On SUN, counts in the MB ranges in general can be attributed to disk accesses and counts in the KB ranges can be attributed to terminal IO. This is useful when looking for processes with heavy disk IO activity. This may vary depending on the sample interval length.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_IPC_SUBSYSTEM_WAIT_PCT
THREAD_IPC_SUBSYSTEM_WAIT_PCT

The percentage of time the process or kernel thread was blocked on the InterProcess Communication (IPC) subsystems (waiting for its interprocess communication activity to complete) during the interval. This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_IPC_SUBSYSTEM_WAIT_PCT_CUM
THREAD_IPC_SUBSYSTEM_WAIT_PCT_CUM

The percentage of time process or kernel thread was blocked on the InterProcess Communication (IPC) subsystems (waiting for its interprocess communication activity to complete) over the cumulative collection time. This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_IPC_SUBSYSTEM_WAIT_TIME THREAD_IPC_SUBSYSTEM_WAIT_TIME

The time, in seconds, the process or kernel thread was blocked on the InterProcess Communication (IPC) subsystems (waiting for its interprocess communication activity to complete) during the interval. This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_IPC_SUBSYSTEM_WAIT_TIME_CUM THREAD_IPC_SUBSYSTEM_WAIT_TIME_CUM

The time, in seconds, the process or kernel thread was blocked on the InterProcess Communication (IPC) subsystems (waiting for its interprocess communication activity to complete) over the cumulative collection time. This is the sum of processes or kernel threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_IPC_WAIT_PCT
THREAD_IPC_WAIT_PCT

The percentage of time the process or kernel thread was blocked onIPC (waiting for interprocess communication calls to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_IPC_WAIT_PCT_CUM
THREAD_IPC_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked onIPC waiting for interprocess communication calls to complete over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_IPC_WAIT_TIME
THREAD_IPC_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked onInterProcess Communication (IPC) (waiting for its interprocess communication calls to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that

have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_IPC_WAIT_TIME_CUM
THREAD_IPC_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked onInterProcess Communication (IPC) (waiting for its interprocess communication calls to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_JOBCTL_WAIT_PCT
THREAD_JOBCTL_WAIT_PCT

The percentage of time during the interval the process or kernel thread was blocked on job control (having been stopped with the job control facilities) during the interval. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_JOBCTL_WAIT_PCT_CUM THREAD_JOBCTL_WAIT_PCT_CUM

The percentage of time the process or kernel thread was blocked on job control (having been stopped with the job control facilities) over the cumulative collection time. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_JOBCTL_WAIT_TIME THREAD_JOBCTL_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on job control (having been stopped with the job control facilities) during the interval. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_JOBCTL_WAIT_TIME_CUM THREAD_JOBCTL_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on job control (having been stopped with the job control facilities) over the cumulative collection time. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_LAN_WAIT_PCT
THREAD_LAN_WAIT_PCT

The percentage of time the process or kernel thread was blocked on LAN (waiting for IO over the LAN to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_LAN_WAIT_PCT_CUM
THREAD_LAN_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on LAN (waiting for IO over the LAN to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_LAN_WAIT_TIME
THREAD_LAN_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on LAN (waiting for IO over the LAN to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_LAN_WAIT_TIME_CUM
THREAD_LAN_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on LAN (waiting for IO over the LAN to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_LDOM_COUNT

The number of Locality Domains the process can potentially obtain memory from.

PROC_LDOM_ID

The identifier for the Locality Domain. This identifier is 'na' for global and cross-LDOM memory.

PROC_LDOM_PCT

The percentage of memory this process utilizes in this Locality Domain.

PROC_LDOM_PRIVATE

The amount of private resident memory the process utilizes in this Locality Domain.

PROC_LDOM_SHARED

The amount of shared resident memory the process utilizes in this Locality Domain.

PROC_LDOM_SUM_PRIVATE

The amount of private resident memory the process utilizes across all Locality Domains.

PROC_LDOM_SUM_SHARED

The amount of shared resident memory the process utilizes across all Locality Domains.

PROC_LDOM_SUM_TOTAL

The total amount of resident memory the process utilizes across all Locality Domains.

PROC_LDOM_SUM_WEIGHTED

The total amount of resident memory the process utilizes across all Locality Domains, taking into account the number of references for shared pages.

This amount is weighted by the number of references. For example, if the process regions include 100Mb of shared pages and these pages are shared with a single other process, the weighted count will be 50Mb for each process.

This is a more realistic indicator of the actual memory the process utilizes.

PROC_LDOM_TOTAL

The total amount of resident memory the process utilizes in this Locality Domain.

PROC_LDOM_TYPE

PROC_LDOM_WEIGHTED

The total amount of resident memory the process utilizes in this Locality Domain, taking into account the number of references for shared pages.

This amount is weighted by the number of references. For example, if the process regions include 100Mb of shared pages and these pages are shared with a single other process, the weighted count will be 50Mb for each process.

This is a more realistic indicator of the actual memory the process utilizes.

PROC_MAJOR_FAULT
THREAD_MAJOR_FAULT

Number of major page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

On HP-UX, major page faults and minor page faults are a subset of vfaults (virtual faults). Stack and heap accesses can cause vfaults, but do not result in a disk page having to be loaded into memory.

PROC_MAJOR_FAULT_CUM
THREAD_MAJOR_FAULT_CUM

Number of major page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, major page faults and minor page faults are a subset of vfaults (virtual faults). Stack and heap accesses can cause vfaults, but do not result in a disk page having to be loaded into memory.

PROC_MEM_PRIVATE_RES
THREAD_MEM_PRIVATE_RES

The size (in KB) of resident memory of private regions only, such as data and stack, currently consumed by this process.

On HP-UX, this metric is initialized only when the menu option “Process Memory Region” is activated for the process. A value of “na” is displayed otherwise.

A value of “na” is displayed when this information is unobtainable.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_MEM_RES
THREAD_MEM_RES

The size (in KB) of resident memory allocated for the process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, the calculation of this metric differs depending on whether this process has used any CPU time since the midaemon process was started. This metric is less accurate and does not include shared memory regions in its calculation when the process has been idle since the midaemon was started.

On HP-UX, for processes that use CPU time subsequent to midaemon startup, the resident memory is calculated as

```
RSS = sum of private region pages +
       (sum of shared region pages /
        number of references)
```

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

This value is only updated when a process uses CPU. Thus, under memory pressure, this value may be higher than the actual amount of resident memory for processes which are idle because their memory pages may no longer be resident or the reference count for shared segments may have changed.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

A value of “na” is displayed when this information is unobtainable. This information may not be obtainable for some system (kernel) processes. It may also not be available for <defunct> processes.

On AIX, this is the same as the RSS value shown by “ps v”.

On Windows, this is the number of KBs in the working set of this process. The working set includes the memory pages touched recently by the threads of the process. If free memory in the system is above a threshold, then pages are left in the working set even if they are not in use. When free memory falls below a threshold, pages are trimmed from the working set, but not necessarily paged out to disk from memory. If those pages are subsequently referenced, they will be page faulted back into the working set. Therefore, the working set is a general indicator of the memory resident set size of this process, but it will vary depending on the overall status of memory on the system. Note that the size of the working set is often larger than the amount of pagefile space consumed (PROC_MEM_VIRT).

PROC_MEM_RES_HIGH
THREAD_MEM_RES_HIGH

The largest value of resident memory (in KB) during its lifetime.

See the description for PROC_MEM_RES for details about how resident memory is determined.

A value of “na” is displayed when this information is unobtainable.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_MEM_SHARED_RES
THREAD_MEM_SHARED_RES

The size (in KB) of resident memory of shared regions only, such as shared text, shared memory, and shared libraries.

On HP-UX, this value is not affected by the reference count. A value of “na” is displayed when this information is unobtainable.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_MEM_VFAULT_COUNT
THREAD_MEM_VFAULT_COUNT

The number of times the CPU handled vfaults on behalf of this process or kernel thread during the interval. On HP-UX, major page faults and minor page faults are a subset of vfaults (virtual faults). Stack and heap accesses can cause vfaults, but do not result in a disk page having to be loaded into memory.

On HP-UX, all exceptions (including faults) cause traps. These include pfaults (protection faults), vfaults (virtual faults), time slice expiration (rescheduling), zero divide, illegal or privileged instructions, single-stepping, breakpoints, and so on. The kernel trap handler code will switch trap counters for vfaults and pfaults to fault counters when appropriate. As such, the trap count excludes vfaults and pfaults.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_MEM_VFAULT_COUNT_CUM
THREAD_MEM_VFAULT_COUNT_CUM

The number of times the CPU handled vfaults on behalf of this process or kernel thread over the cumulative collection time. On HP-UX, major page faults and minor page faults are a subset of vfaults (virtual faults). Stack and heap accesses can cause vfaults, but do not result in a disk page having to be loaded into memory.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, all exceptions (including faults) cause traps. These include pfaults (protection faults), vfaults (virtual faults), time slice expiration (rescheduling), zero divide, illegal or privileged instructions, single-stepping, breakpoints, and so on. The kernel trap handler code will switch trap counters for vfaults and pfaults to fault counters when appropriate. As such, the trap count excludes vfaults and pfaults.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_MEM_VIRT
THREAD_MEM_VIRT

The size (in KB) of virtual memory allocated for the process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, this consists of the sum of the virtual set size of all private memory regions used by this process, plus this process' share of memory regions which are shared by multiple processes. For processes that use CPU time, the value is divided by the reference count for those regions which are shared.

On HP-UX, this metric is less accurate and does not reflect the reference count for shared regions for processes that were started prior to the midaemon process and have not used any CPU time since the midaemon was started.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

On all other Unix systems, this consists of private text, private data, private stack and shared memory. The reference count for shared memory is not taken into account, so the value of this metric represents the total virtual size of all regions regardless of the number of processes sharing access.

Note also that lazy swap algorithms, sparse address space malloc calls, and memory-mapped file access can result in large VSS values. On systems that provide Glance memory regions detail reports, the drilldown detail per memory region is useful to understand the nature of memory allocations for the process.

A value of “na” is displayed when this information is unobtainable. This information may not be obtainable for some system (kernel) processes. It may also not be available for <defunct> processes.

On Windows, this is the number of KBs the process has used in the paging file(s). Paging files are used to store pages of memory used by the process, such as local data, that are not contained in other files. Examples of memory pages which are contained in other files include pages storing a program's .EXE and .DLL files. These would not be kept in pagefile space. Thus, often programs will have a memory working set size (PROC_MEM_RES) larger than the size of its pagefile space.

On Linux this value is rounded to PAGESIZE.

PROC_MEM_WAIT_PCT
THREAD_MEM_WAIT_PCT

The percentage of time the process or kernel thread was blocked on memory (waiting for memory resources to become available) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on

Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_MEM_WAIT_PCT_CUM THREAD_MEM_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on memory (waiting for memory resources to become available) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_MEM_WAIT_TIME THREAD_MEM_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on VM (waiting for virtual memory resources to become available) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_MEM_WAIT_TIME_CUM THREAD_MEM_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on VM (waiting for virtual memory resources to become available) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_MINOR_FAULT
THREAD_MINOR_FAULT

Number of minor page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

On HP-UX, major page faults and minor page faults are a subset of vfaults (virtual faults). Stack and heap accesses can cause vfaults, but do not result in a disk page having to be loaded into memory.

PROC_MINOR_FAULT_CUM
THREAD_MINOR_FAULT_CUM

Number of minor page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, major page faults and minor page faults are a subset of vfaults (virtual faults). Stack and heap accesses can cause vfaults, but do not result in a disk page having to be loaded into memory.

PROC_MSG_RECEIVED
THREAD_MSG_RECEIVED

The number of socket messages received by a process or kernel thread during the interval. This does not include SYSV messages (msgrcv).

PROC_MSG_RECEIVED_CUM
THREAD_MSG_RECEIVED_CUM

The total number of socket messages received by a process or kernel thread over the cumulative collection time. This does not include SYSV messages (msgrcv).

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

PROC_MSG_SENT
THREAD_MSG_SENT

The number of socket messages sent by a process or kernel thread during the interval. This does not include SYSV messages (msgsnd).

PROC_MSG_SENT_CUM
THREAD_MSG_SENT_CUM

The total number of socket messages sent by a process or kernel thread over the cumulative collection time. This does not include SYSV messages (msgsnd).

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

PROC_MSG_WAIT_PCT
THREAD_MSG_WAIT_PCT

The percentage of time the process or kernel thread was blocked on messages (waiting for message queue operations to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_MSG_WAIT_PCT_CUM
THREAD_MSG_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on messages (waiting for message queue operations to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_MSG_WAIT_TIME
THREAD_MSG_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on messages (waiting for message queue operations to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_MSG_WAIT_TIME_CUM
THREAD_MSG_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on messages (waiting for message queue operations to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_NFS_WAIT_PCT
THREAD_NFS_WAIT_PCT

The percentage of time the process or kernel thread was blocked on NFS (waiting for network file system IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_NFS_WAIT_PCT_CUM
THREAD_NFS_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on NFS (waiting for network file system IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_NFS_WAIT_TIME
THREAD_NFS_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on NFS (waiting for its network file system IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_NFS_WAIT_TIME_CUM
THREAD_NFS_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on NFS (waiting for its network file system IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_NICE_PRI
THREAD_NICE_PRI

The nice priority for the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) when it was last dispatched. The value is a bias used to adjust the priority for the process.

On AIX, the nice user value, makes a process less favored than it otherwise would be, has a range of 0-40 with a default value of 20. The value of PUSER is always added to the value of nice to weight the user process down below the range of priorities expected to be in use by system jobs like the scheduler and special wait queues.

On all other Unix systems, the value ranges from 0 to 39. A higher value causes a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) to be dispatched less.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_NONDISK_LOGL_READ THREAD_NONDISK_LOGL_READ

The number of non-disk logical reads (that is, calls to read(2)) made by a process or kernel thread during the interval.

“Non-disk” devices include terminals, tapes, and so forth on the local or remote machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_NONDISK_LOGL_READ_CUM THREAD_NONDISK_LOGL_READ_CUM

The number of non-disk logical reads (that is, calls to read(2)) made by a process or kernel thread over the cumulative collection time.

“Non-disk” devices include terminals, tapes, and so forth on the local or remote machine.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_NONDISK_LOGL_WRITE
THREAD_NONDISK_LOGL_WRITE

The number of non-disk logical writes (that is, calls to write(2)) made by a process or kernel thread during the interval.

“Non-disk” devices include terminals, tapes, and so forth on the local or remote machine.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_NONDISK_LOGL_WRITE_CUM
THREAD_NONDISK_LOGL_WRITE_CUM

The number of non-disk logical writes (that is, calls to write(2)) made by a process or kernel thread over the cumulative collection time.

“Non-disk” devices include terminals, tapes, and so forth on the local or remote machine.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_NONDISK_PHYS_READ
THREAD_NONDISK_PHYS_READ

The number of physical non-disk reads made by a process or kernel thread during the interval to buffered/block devices, such as a tape drive.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_NONDISK_PHYS_READ_CUM THREAD_NONDISK_PHYS_READ_CUM

The number of local/remote physical non-disk reads made by a process or kernel thread over the cumulative collection time to buffered/block devices, such as a tape drive.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_NONDISK_PHYS_WRITE THREAD_NONDISK_PHYS_WRITE

The number of local/remote physical non-disk writes made by a process or kernel thread during the interval to buffered/block devices, such as a tape drive.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_NONDISK_PHYS_WRITE_CUM THREAD_NONDISK_PHYS_WRITE_CUM

Number of local/remote physical non-disk writes made by a process or kernel thread over the cumulative collection time to buffered/block devices, such as a tape drive.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_OPEN THREAD_OPEN

The number of file, socket, or pipe opens made by the process or kernel thread during the interval. This corresponds to the number of open(2) system calls.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_OPEN_CUM THREAD_OPEN_CUM

The number of file, socket, or pipe opens made by the process or kernel thread over the cumulative collection time. This corresponds to the number of open(2) system calls.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_OTHER_IO_WAIT_PCT
THREAD_OTHER_IO_WAIT_PCT

The percentage of time the process or kernel thread was blocked on “other IO” during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_OTHER_IO_WAIT_PCT_CUM THREAD_OTHER_IO_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on “other IO” over the cumulative collection time. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_OTHER_IO_WAIT_TIME
THREAD_OTHER_IO_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on other IO during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_OTHER_IO_WAIT_TIME_CUM
THREAD_OTHER_IO_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on “other IO” over the cumulative collection time. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_OTHER_WAIT_PCT
THREAD_OTHER_WAIT_PCT

The percentage of time the process or kernel thread was blocked on other (unknown) activities during the interval. This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_OTHER_WAIT_PCT_CUM
THREAD_OTHER_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on other (unknown) activities over the cumulative collection time. This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_OTHER_WAIT_TIME
THREAD_OTHER_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on other (unknown) activities during the interval. This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_OTHER_WAIT_TIME_CUM
THREAD_OTHER_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on other (unknown) activities over the cumulative collection time. This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_PAGEFAULT
THREAD_PAGEFAULT

The number of page faults that occurred during the interval for the process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

PROC_PAGEFAULT_RATE
THREAD_PAGEFAULT_RATE

The number of page faults per second that occurred during the interval for the process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

PROC_PAGEFAULT_RATE_CUM
THREAD_PAGEFAULT_RATE_CUM

The average number of page faults per second that occurred over the cumulative collection time for the process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

PROC_PARENT_PROC_ID
THREAD_PARENT_PROC_ID

The parent process' PID number.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_PIPE_WAIT_PCT
THREAD_PIPE_WAIT_PCT

The percentage of time the process or kernel thread was blocked onPIPE (waiting for pipe communication to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_PIPE_WAIT_PCT_CUM
THREAD_PIPE_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked onPIPE (waiting for pipe communication to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_PIPE_WAIT_TIME THREAD_PIPE_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked onPIPE (waiting for pipe communication to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_PIPE_WAIT_TIME_CUM
THREAD_PIPE_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked onPIPE (waiting for pipe communication to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_PRI
THREAD_PRI

On Unix systems, this is the dispatch priority of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) at the end of the interval. The lower the value, the more likely the process is to be dispatched.

On Windows, this is the current base priority of this process.

On HP-UX, whenever the priority is changed for the selected process or kernel thread, the new value will not be reflected until the process or kernel thread is reactivated if it is currently idle (for example, SLEEPing).

On HP-UX, the lower the value, the more the process or kernel thread is likely to be dispatched. Values between zero and 127 are considered to be “real-time” priorities, which the kernel does not adjust. Values above 127 are normal priorities and are modified by the kernel for load balancing. Some special priorities are used in the HP-UX kernel and subsystems for different activities. These values are described in /usr/include/sys/param.h. Priorities less than PZERO 153 are not signalable.

Note that on HP-UX, many network-related programs such as inetd, biod, and rlogind run at priority 154 which is PPIPE. Just because they run at this priority does not mean they are using pipes. By examining the open files, you can determine if a process or kernel thread is using pipes.

For HP-UX 10.0 and later releases, priorities between -32 and -1 can be seen for processes or kernel threads using the Posix Real-time Schedulers. When specifying a Posix priority, the value entered must be in the range from 0 through 31, which the system then remaps to a negative number in the range of -1 through -32. Refer to the rtsched man pages for more information.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

On AIX, values for priority range from 0 to 127. Processes running at priorities less than PZERO (40) are not signalable.

On Windows, the higher the value the more likely the process or thread is to be dispatched. Values for priority range from 0 to 31. Values of 16 and above are considered to be “realtime” priorities. Threads within a process can raise and lower their own base priorities relative to the process's base priority.

PROC_PRI_WAIT_PCT
THREAD_PRI_WAIT_PCT

The percentage of time during the interval the process or kernel thread was blocked on priority (waiting for its priority to become high enough to get the CPU).

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_PRI_WAIT_PCT_CUM
THREAD_PRI_WAIT_PCT_CUM

The percentage of time the process or kernel thread was blocked on priority over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_PRI_WAIT_TIME
THREAD_PRI_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on PRI (waiting for its priority to become high enough to get the CPU) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_PRI_WAIT_TIME_CUM
THREAD_PRI_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on PRI (waiting for its priority to become high enough to get the CPU) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_PRMID
THREAD_PRMID

The PRM Group ID this process is assigned to. The PRM group configuration is kept in the PRM configuration file.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_PROC_ARGV1
THREAD_PROC_ARGV1

The first argument (argv[1]) of the process argument list or the second word of the command line, if present. (For kernel threads, if HP-UX/Linux Kernel 2.6 and above this metric returns the value of the associated process). The HP Performance Agent logs the first 32 characters of this metric.

For releases that support the parm file javaarg flag, this metric may not be the first argument. When javaarg=true, the value of this metric is replaced (for java processes only) by the java class or jar name. This can then be useful to construct parm file java application definitions using the argv1= keyword.

PROC_PROC_CMD
THREAD_PROC_CMD

The full command line with which the process was initiated. (For kernel threads, if HP-UX/Linux Kernel 2.6 and above this metric returns the value of the associated process).

On HP-UX, the maximum length returned depends upon the version of the OS, but typically up to 1020 characters are available.

On other Unix systems, the maximum length is 4095 characters.

On Linux, if the command string exceeds 4096 characters, the kernel instrumentation may not report any value.

If the command line contains special characters, such as carriage return and tab, these characters will be converted to , , and so on.

PROC_PROC_ID
THREAD_PROC_ID

The process ID number (or PID) of this process(or associated process for kernel threads, if HPUX/LInux Kernel 2.6 and above) that is used by the kernel to uniquely identify the process. Process numbers are reused, so they only identify a process for its lifetime.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_PROC_NAME
THREAD_PROC_NAME

The process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above) program name. It is limited to 16 characters.

On Unix systems, this is derived from the 1st parameter to the exec(2) system call.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

On Windows, the “System Idle Process” is not reported by Perf Agent since Idle is a process that runs to occupy the processors when they are not executing other threads. Idle has one thread per processor.

PROC_REGION_FILENAME

The file path that corresponds to the front store file of a memory region. For text and data regions, this is the name of the program; for shared libraries it is the library name.

Certain “special” names are displayed if there is no actual “front store” for a memory region. These special names correspond to the region type (for example, <stack>). If the name is “<mmap>”, then this is a memory region without “front store,” created by the system call mmap(2).

If the file format includes an inode number, use the program ncheck (1M) to display the filename relative to the mount point. Sometimes files may be deleted before they are closed. In these cases, the process file table may still have the inode even though the file is not actually present and as a result, ncheck will fail.

In the following example, note that the file system name has been included to avoid the overhead of searching all of the file systems for the inode number.

If the following file name was displayed:

```
<vxfs,/,/dev/root,inode:926>
```

and then from that display, the following ncheck command was entered:

```
ncheck -i 926 -F vxfs /dev/root
```

An output like the following would be generated:

```
/dev/root:
926     /etc/utmpx
```

The string for an inode is as follows:

```
<www,xxx,yyy,zzz,inode:nnnn>
```

where:

```
www:  Is the file type:
  “reg”  - Regular file
  “dir”  - Directory file
  “blk”  - Block device
  “chr”  - Character device
  “lnk”  - Soft file link
  “sock” - Socket
```

```
      "fifo" - FIFO

    xxx:  Is the file domain; such as,
      "ufs"  - Unix file system
      "nfs"  - NFS
      "vxfs" - Veritas file system

    yyy:  Is the mount point.
    zzz:  Is the file system.
```

On HP-UX 11.0, the file type is not shown since it will always be "reg" for regular files, which are mmappable.

If a program is "hard linked" (that is, two directories pointing to the same inode), then a different name may be reported for the text and data regions than is actually running. This often happens with the HPTERM program, which is often hard-linked to HELPVIEW. Use the "-i" option of the "ls" command to see the inode numbers.

PROC_REGION_LOCKED

The amount of memory (in KBs unless otherwise indicated) that is locked. Memory is typically "locked" by calls to plock(2), datalock(3C), or shmctl(2). In addition to the number of pages locked, this metric includes the number of bytes (rounded up to the nearest page) used by the kernel to store the virtual memory structures allocated to track the pages in the memory region. Hence if the region pages are locked in memory, the virtual memory management structures for that region must also be locked. As a result, one could see the number of bytes locked exceed the virtual memory size of a region.

This metric is currently unavailable on HP-UX 11.0.

PROC_REGION_PAGE_COUNT_1_4KB

The number of pages of size 4KB allocated to this memory region.

PROC_REGION_PAGE_COUNT_2_16KB

The number of pages of size 16KB allocated to this memory region.

PROC_REGION_PAGE_COUNT_3_64KB

The number of pages of size 64KB allocated to this memory region.

PROC_REGION_PAGE_COUNT_4_256KB

The number of pages of size 256KB allocated to this memory region.

PROC_REGION_PAGE_COUNT_5_1MB

The number of pages of size 1MB allocated to this memory region.

PROC_REGION_PAGE_COUNT_6_4MB

The number of pages of size 4MB allocated to this memory region.

PROC_REGION_PAGE_COUNT_7_16MB

The number of pages of size 16MB allocated to this memory region.

PROC_REGION_PAGE_COUNT_8_64MB

The number of pages of size 64MB allocated to this memory region.

PROC_REGION_PAGE_COUNT_9_256MB

The number of pages of size 256MB allocated to this memory region. Even though 11.11 and later releases support page sizes larger than 64Mb, only 11.23 and newer releases report this data. Versions older than 11.23 adjust 64Mb page counts to include larger page sizes (for instance, one 256Mb page will be counted as 4 64Mb pages).

PROC_REGION_PAGE_COUNT_B_1GB

The number of pages of size 1GB allocated to this memory region. Even though 11.11 and later releases support page sizes larger than 64Mb, only 11.23 and newer releases report this data. Versions older than 11.23 adjust 64Mb page counts to include larger page sizes (for instance, one 256Mb page will be counted as 4 64Mb pages).

PROC_REGION_PAGE_COUNT_B_4GB

The number of pages of size 4GB allocated to this memory region. Even though 11.11 and later releases support page sizes larger than 64Mb, only 11.23 and newer releases report this data. Versions older than 11.23 adjust 64Mb page counts to include larger page sizes (for instance, one 256Mb page will be counted as 4 64Mb pages).

PROC_REGION_PAGE_SIZE_HINT

The recommended or default size for pages allocated to this memory region. The chatr(1) command can be used to change the page size hint requested for a program's text and data regions.

PROC_REGION_PRIVATE_SHARED_FLAG

A text indicator of either private memory (Priv) or shared (Shared) for this memory region. Private memory is only being used by the current process. Shared memory is mapped into the address space of other processes.

PROC_REGION_REF_COUNT

The number of processes sharing this memory region.

For private regions this value is 1. For shared regions, this value is the number of processes sharing the region.

This metric is currently unavailable on HP-UX 11.0.

PROC_REGION_RES

The size (in KBs unless otherwise indicated) of the resident memory occupied by this memory region.

On HP-UX 11.0 and beyond, this value is not affected by the reference count.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

PROC_REGION_RES_DATA

The size (in KBs unless otherwise indicated) of the total resident memory occupied by data regions of this process. This value is not affected by the reference count since all data regions are private.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

PROC_REGION_RES_OTHER

The size (in KBs unless otherwise indicated) of the total resident memory occupied by regions of this process that are not text, data, stack, or shared memory.

On HP-UX 11.0 and beyond, this value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

PROC_REGION_RES_SHMEM

The size (in KBs unless otherwise indicated) of the total resident memory occupied by shared memory regions of this process.

On HP-UX 11.0 and beyond, this value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

In other words, the sum of this value for all processes exceeds the actual memory occupied since some memory is shared.

PROC_REGION_RES_STACK

The size (in KBs unless otherwise indicated) of the total resident memory occupied by stack regions of this process.

On HP-UX, stack regions are always private and will have a reference count of one. The stack, in this case, refers to the kernel stack, not the user stack. For a threaded OS, each kernel thread will have one kernel stack.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

PROC_REGION_RES_TEXT

The size (in KBs unless otherwise indicated) of the total resident memory occupied by text regions of this process.

On HP-UX 11.0 and beyond, this value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

In other words, the sum of this value for all processes exceeds the actual memory occupied since some memory is shared.

PROC_REGION_TYPE

A text name for the type of this memory region. It can be one of the following:

DATA	Data region
LIBDAT	Shared Library data
LIBTXT	Shared Library text
STACK	Stack region
TEXT	Text (that is, code)

On HP-UX, it can also be one of the following:

GRAPH	Frame buffer lock page
IOMAP	IO region (iomap)
MEMMAP	Memory-mapped file, which includes shared libraries (text and data), or memory created by calls to mmap(2)
NULLDR	Null pointer dereference shared page (see below)
RSESTA	Itanium Registered stack engine region
SIGSTK	Signal stack region
UAREA	User Area region
UNKNWN	Region of unknown type

On HP-UX, a whole page is allocated for NULL pointer dereferencing, which is reported as the NULLDR area. If the program is compiled with the "-z" option (which disallows NULL dereferencing), this area is missing. Shared libraries are accessed as memory mapped files, so that the code will show up as "MEMMAP/Shared" and data will show up as "MEMMAP/Priv".

On SUN, it can also be one of the following:

BSS	Static initialized data
MEMMAP	Memory mapped files
NULLDR	Null pointer dereference shared page (see below).
SHMEM	Shared memory
UNKNWN	Region of unknown type

On SUN, programs might have an area for NULL pointer dereferencing, which is reported as the NULLDR area. Special segment types that are supported by the kernel that are used for frame buffer devices or other purposes are typed as UNKNWN. The following kernel processes are examples of this: sched, pageout, and fsflush.

On AIX, as of mid-2010, the OS only provides information for text and data.

PROC_REGION_VIRT

The size (in KBs unless otherwise indicated) of the virtual memory occupied by this memory region.

This value is not affected by the reference count.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_ADDRS

The virtual address of this memory region displayed in hexadecimal showing the space and offset of the region.

On HP-UX, this is a 64-bit (96-bit on a 64-bit OS) hexadecimal value indicating the space and space offset of the region.

PROC_REGION_VIRT_DATA

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by data regions of this process. This value is not affected by the reference count since all data regions are private.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_OTHER

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by regions of this process that are not text, data, stack, or shared memory.

This value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_SHMEM

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by shared memory regions of this process.

Note that this memory is shared by other processes and this figure is reported in their metrics also.

This value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_STACK

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by stack regions of this process.

Stack regions are always private and will have a reference count of one.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_TEXT

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by text regions of this process. This value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_RPC_WAIT_PCT
THREAD_RPC_WAIT_PCT

The percentage of time the process or kernel thread was blocked on RPC (waiting for remote procedure calls to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_RPC_WAIT_PCT_CUM
THREAD_RPC_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on RPC (waiting for remote procedure calls to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_RPC_WAIT_TIME
THREAD_RPC_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on RPC (waiting for its remote procedure calls to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_RPC_WAIT_TIME_CUM
THREAD_RPC_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on RPC (waiting for its remote procedure calls to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_RUN_TIME
THREAD_RUN_TIME

The elapsed time since a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) started, in seconds.

This metric is less than the interval time if the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was not alive during the entire first or last interval.

On a threaded operating system such as HP-UX 11.0 and beyond, this metric is available for a process or kernel thread.

PROC_SCHEDULER
THREAD_SCHEDULER

The scheduling policy for this process or kernel thread.

The available scheduling policies are:

```
HPUX   - HP-UX normal timeshare
NOAGE  - HP-UX timeshare without
         usage decay
RTPRIO - HP-UX Real-time
FIFO   - Posix First In/First Out
RR     - Posix Round-Robin
```

```
RR2     - Posix Round-Robin with a
          per-priority time slice
          interval
```

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

PROC_SEM_WAIT_PCT
THREAD_SEM_WAIT_PCT

The percentage of time the process or kernel thread was blocked onsemaphores (waiting on a semaphore operation to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SEM_WAIT_PCT_CUM
THREAD_SEM_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked onsemaphores (waiting on a semaphore operation to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process

collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SEM_WAIT_TIME
THREAD_SEM_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked onsemaphores (waiting on a semaphore operation to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_SEM_WAIT_TIME_CUM
THREAD_SEM_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked onsemaphores (waiting on a semaphore operation to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_SIGNAL
THREAD_SIGNAL

Number of signals seen by the current process (or kernel thread, if HP-UX) during the lifetime of the process or kernel thread.

PROC_SIGNAL_CUM
THREAD_SIGNAL_CUM

Number of signals seen by the current process (or kernel thread, if HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

PROC_SLEEP_WAIT_PCT
THREAD_SLEEP_WAIT_PCT

The percentage of time the process or kernel thread was blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SLEEP_WAIT_PCT_CUM
THREAD_SLEEP_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on SLEEP (waiting to awaken from sleep system calls) over the cumulative collection time. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SLEEP_WAIT_TIME
THREAD_SLEEP_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that

have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_SLEEP_WAIT_TIME_CUM
THREAD_SLEEP_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on SLEEP (waiting to awaken from sleep system calls) over the cumulative collection time. A process or kernel thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_SOCKET_WAIT_PCT
THREAD_SOCKET_WAIT_PCT

The percentage of time the process or kernel thread was blocked on sockets (waiting for their IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SOCKET_WAIT_PCT_CUM THREAD_SOCKET_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on sockets (waiting for their IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on

Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SOCKET_WAIT_TIME THREAD_SOCKET_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on sockets (waiting for its IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_SOCKET_WAIT_TIME_CUM THREAD_SOCKET_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on sockets (waiting for its IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that

have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_STARTTIME
THREAD_STARTTIME

The creation date and time of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

PROC_STATE
THREAD_STATE

A text string summarizing the current state of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above), either:

```
new      This is the first interval
         the process has been
         displayed.
active   Process is continuing.
died     Process expired during
         the interval.
```

PROC_STOP_REASON
THREAD_STOP_REASON

A text string describing what caused the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) to stop executing. For example, if the process is waiting for a CPU while higher priority processes are executing, then its block reason is PRI. A complete list of block reasons follows:

```
String   Reason for Process Block
------------------------------------

CACHE   Waiting at the buffer cache
        level trying to lock down a
        buffer cache structure, or
        waiting for an IO operation
        to or from a buffer cache to
        complete. File system access
        will block on IO more often
        than CACHE on HP-UX 11.x.

CDFS    Waiting for CD-ROM file
        system node structure
        allocation or locks while
```

	accessing a CD-ROM device through the file system.
died	Process terminated during the interval.
DISK	Waiting for an IO operation to complete at the logical device manager or disk driver level. Waits from raw disk IO and diagnostic requests can be seen here. Buffered IO requests can also block on DISK, but will more often be seen waiting on “IO”. CDFS access will block on “CDFS”. Virtual memory activity will block on “VM”.
GRAPH	Waiting for a graphics card or framebuf semaphore operation.
INODE	Waiting while accessing an inode structure. This includes inode gets and waiting due to inode locks.
IO	Waiting for IO to local disks, printers, tapes, or instruments to complete (above the driver, but below the buffer cache). Both file system and raw disk access can block in this state. CDFS access will block on “CDFS”. Virtual memory activity will block on “VM”.
IPC	Waiting for a process or kernel thread event (that is, waiting for a child to receive a signal). This includes both inter and intra process or kernel thread operations, such as IPC locks, kernel thread mutexes, and database IPC operations. System V

```
        message queue operations
        will block on “MESG”, while
        semaphore operations will
        block on “SEM”.

JOBCL   Waiting for tracing resume,
        debug resume, or job control
        start.  A background process
        incurs this block when
        attempting to write to a
        terminal set with “stty
        tostop‟.  On HP-UX 11i,
        scheduler activation threads
        (user threads) will show
        this block.

LAN     Waiting for a network IO
        completion.  This includes
        waiting on the LAN hardware
        and low level LAN device
        driver.  It does not include
        waiting on the higher level
        network software such as the
        streams based transport or
        NFS, which has its own stop
        state.

MESG    Waiting for a System V
        message queue operation such
        as msgrcv or msgsnd.

new     Process was created (via the
        fork/vfork system calls)
        during the interval.

NFS     Waiting for a Networked File
        System request to complete.
        This includes both NFS V2
        and V3 requests.  This does
        not include stops where
        kernel threads or deamons
        are waiting for a NFS event
        or request (such as biod or
        nfsd).  These will block on
        SLEEP to show they are
        waiting for some activity.

NONE    Zombie process - waiting to
        die.
```

OTHER	The process was started before the midaemon was started and has not been resumed, or the block state is unknown.
PIPE	Waiting for operations involving pipes. This includes opening, closing, reading, and writing using pipes. Named pipes will block on PIPE.
PRI	Waiting because a higher priority process is running, or waiting for a spinlock or alpha semaphore.
RPC	Waiting for remote procedure call operations to complete. This includes both NFS and DCE RPC requests.
SEM	Waiting for a System V semaphore operation (such as semop, semget, or semctl) or waiting for a memory mapped file semaphore operation (such as msem_init or msem_lock).
SLEEP	Waiting because the process put itself to sleep using system calls such as sleep, wait, pause, sigpause, poll, sigsuspend and select. This is the standard stop reason for idle system daemons.
SOCKT	Waiting for an operation to complete while accessing a device through a socket. This is used primarily in networking code and includes all protocols using sockets (X25, UDP, TCP, and so on).
STRMS	Waiting for an operation to complete while accessing a “streams” device. This is

```
        the normal stop reason for
        kernel threads and daemons
        waiting for a streams event.
        This includes the network
        transport and pseudo
        terminal IO requests.  For
        example, waiting for a read
        on a streams device or
        waiting for an internal
        streams synchronization.

SYSTM   Waiting for access to a
        system resource or lock.
        These resources include data
        structures from the LVM,
        VFS, UFS, JFS, and Disk
        Quota subsystems.  "SYSTM"
        is the "catch-all" wait
        state for blocks on system
        resources that are not
        common enough or long enough
        to warrant their own stop
        state.

TERM    Waiting for a non-streams
        terminal transfer (tty or
        pty).

VM      Waiting for a virtual memory
        operation to complete, or
        waiting for free memory, or
        blocked while creating/
        accessing a virtual memory
        structure.
```

For a process or kernel thread currently running, the last reason it was stopped before obtaining the CPU is shown.

On HP-UX 11.0 and beyond, mikslp.text (located in /opt/perf/lib) contains the blocking functions and their corresponding block states for use by midaemon.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

PROC_STOP_REASON_FLAG
THREAD_STOP_REASON_FLAG

A numeric value for the stop reason. This is used by scopeux instead of the ASCII string returned by PROC_STOP_REASON in order to conserve space in the log file.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

PROC_STREAM_WAIT_PCT
THREAD_STREAM_WAIT_PCT

The percentage of time the process or kernel thread was blocked on streams IO (waiting for a streams IO operation to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_STREAM_WAIT_PCT_CUM
THREAD_STREAM_WAIT_PCT_CUM

The average percentage of time the process or thread was blocked on streams IO (waiting for a streams IO operation to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_STREAM_WAIT_TIME
THREAD_STREAM_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on streams IO (waiting for a streams IO operation to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_STREAM_WAIT_TIME_CUM
THREAD_STREAM_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on streams IO (waiting for a streams IO operation to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_SWAP
THREAD_SWAP

The number of times the process or kernel thread was deactivated during the interval.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_SWAP_CUM
THREAD_SWAP_CUM

The number of times the process or kernel thread was deactivated over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_SYS_WAIT_PCT
THREAD_SYS_WAIT_PCT

The percentage of time the process or kernel thread was blocked on system resources during the interval. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SYS_WAIT_PCT_CUM
THREAD_SYS_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on SYSTM (that is, system resources) over the cumulative collection time. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_SYS_WAIT_TIME
THREAD_SYS_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on SYSTM (that is, system resources) during the interval. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_SYS_WAIT_TIME_CUM
THREAD_SYS_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on SYSTM (that is, system resources) over the cumulative collection time. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_TERM_IO_WAIT_PCT
THREAD_TERM_IO_WAIT_PCT

The percentage of time the process or kernel thread was blocked on terminal IO (waiting for its terminal IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_TERM_IO_WAIT_PCT_CUM
THREAD_TERM_IO_WAIT_PCT_CUM

The average percentage of time the process or kernel thread was blocked on terminal IO (waiting for its terminal IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

A percentage of time spent in a wait state is calculated as the time a kernel thread (or all kernel threads of a process) spent waiting in this state, divided by the alive time of the kernel thread (or all kernel threads of the process) during the interval.

If this metric is reported for a kernel thread, the percentage value is for that single kernel thread. If this metric is reported for a process, the percentage value is calculated with the sum of the wait and alive times of all of its kernel threads.

For example, if a process has 2 kernel threads, one sleeping for the entire interval and one waiting on terminal input for the interval, the process wait percent values will be 50% on Sleep and 50% on Terminal. The kernel thread wait values will be 100% on Sleep for the first kernel thread and 100% on Terminal for the second kernel thread.

For another example, consider the same process as above, with 2 kernel threads, one of which was created half-way through the interval, and which then slept for the remainder of the interval. The other kernel thread was waiting for terminal input for half the interval, then used the CPU actively for the remainder of the interval. The process wait percent values will be 33% on Sleep and 33% on Terminal (each one third of the total alive time). The kernel thread wait values will be 100% on Sleep for the first kernel thread and 50% on Terminal for the second kernel thread.

PROC_TERM_IO_WAIT_TIME
THREAD_TERM_IO_WAIT_TIME

The time, in seconds, that the process or kernel thread was blocked on terminal IO (waiting for its terminal IO to complete) during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_TERM_IO_WAIT_TIME_CUM
THREAD_TERM_IO_WAIT_TIME_CUM

The time, in seconds, that the process or kernel thread was blocked on terminal IO (waiting for its terminal IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_THREAD_COUNT
THREAD_THREAD_COUNT

The total number of kernel threads for the current process.

On Linux systems with Kernel 2.5 and below, every thread has its own process ID so this metric will always be 1.

On Solaris systems, this metric reflects the total number of Light Weight Processes (LWPs) associated with the process.

PROC_THREAD_ID
THREAD_THREAD_ID

The thread ID number of this kernel thread, used to uniquely identify it. On Linux systems this metric shall be available from Linux Kernel 2.6 onwards.

PROC_TIME
THREAD_TIME

The time the data for the process (or kernel threads, if HP-UX/Linux Kernel 2.6 and above) was collected, in local time.

PROC_TOP_CPU_INDEX
THREAD_TOP_CPU_INDEX

The index of the process which consumed the most CPU during the interval. From this index, the process PID, process name, and CPU utilization can be obtained. (Even for kernel threads if HPUX/Linux Kernel 2.6 and above this metric returns the index of the process)

This metric is used by the Performance Tools to index into the Data collection interface's internal table. This is not a metric that will be interesting to Tool users.

PROC_TOP_DISK_INDEX
THREAD_TOP_DISK_INDEX

The index of the process which did the most physical IOs during the last interval.

On HP-UX, note that NFS mounted disks are not considered in this calculation.

With this index, the PID, process name, and IOs per second can be obtained.

This metric is used by the Performance Tools to index into the Data collection interface's internal table. This is not a metric that will be interesting to Tool's users.

PROC_TOTAL_WAIT_TIME
THREAD_TOTAL_WAIT_TIME

The total time, in seconds, that the process or kernel thread spent blocked during the interval.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that

have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_TOTAL_WAIT_TIME_CUM THREAD_TOTAL_WAIT_TIME_CUM

The total time that the process or kernel thread spent blocked over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process wait time is calculated by summing the wait times of its kernel threads. If this metric is reported for a kernel thread, the value is the wait time of that single kernel thread. If this metric is reported for a process, the value is the sum of the wait times of all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. For multi-threaded processes, the wait times can exceed the length of the measurement interval.

PROC_TTY THREAD_TTY

The controlling terminal for a process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above). This field is blank if there is no controlling terminal. On HP-UX, Linux, and AIX, this is the same as the “TTY” field of the ps command.

On all other Unix systems, the controlling terminal name is found by searching the directories provided in the /etc/ttysrch file. See man page ttysrch(4) for details. The matching criteria field (“M”, “F” or “I” values) of the ttysrch file is ignored. If a terminal is not found in one of the ttysrch file directories, the following directories are searched in the order here: “/dev”, “/dev/pts”, “/dev/term” and “dev/xt”. When a match is found in one of the “/dev” subdirectories, “/dev/” is not displayed as part of the terminal name. If no match is found in the directory searches, the major and minor numbers of the controlling terminal are displayed. In most cases, this value is the same as the “TTY” field of the ps command.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_TTY_DEV
THREAD_TTY_DEV

The device number of the controlling terminal for a process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_UID
THREAD_UID

The real UID (user ID number) of a process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above). This is the UID returned from the getuid system call.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_USER_NAME
THREAD_USER_NAME

On Unix systems, this is real user name of a process or the login account (from /etc/passwd) of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above). If more than one account is listed in /etc/passwd with the same user ID (uid) field, the first one is used. If an account cannot be found that matches the uid field, then the uid number is returned. This would occur if the account was removed after a process was started.

On Windows, this is the process owner account name, without the domain name this account resides in.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_USER_THREAD_ID
THREAD_USER_THREAD_ID

The user thread ID number of the last user thread to execute within the context of this process or kernel thread. User threads IDs are used to identify user-level threads of execution within the context of a process. A process may have one or more user threads even if there is only one kernel thread.

PROC_USRPRI
THREAD_USRPRI

The user priority for the process or kernel thread is set by the kernel during scheduling. This value becomes the actual process or kernel thread priority once it returns to user mode from kernel mode.

The calculation of the user priority is based on the process or kernel thread CPU usage and the nice value.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

PROC_VOLUNTARY_CSWITCH
THREAD_VOLUNTARY_CSWITCH

The number of times a process (or kernel thread, if HP-UX) has given up the CPU before an external event preempted it during the interval. Examples of voluntary switches include calls to sleep(2) and select(2).

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

PROC_VOLUNTARY_CSWITCH_CUM
THREAD_VOLUNTARY_CSWITCH_CUM

The number of times a process (or kernel thread, if HP-UX) has given up the CPU before an external event preempted it over the cumulative collection time. Examples of voluntary switches include calls to sleep(2) and select(2).

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

SYSCALL_ACTIVE_CUM

The number of system calls used on the system. All calls used over the cumulative collection time are included in this count.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

SYSCALL_CALL_COUNT

The number of system calls made to this function during the interval.

They are assessed when the system call stub returns control back to the calling program/routine.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

SYSCALL_CALL_COUNT_CUM

The number of system calls made to this function over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

SYSCALL_CALL_ID

The ID number of this system call. System calls are sequentially numbered starting with one.

SYSCALL_CALL_NAME

The system call name.

SYSCALL_CALL_RATE

The number of system calls per second made to this function during the last interval.

They are assessed when the system call stub returns control back to the calling program/routine.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

SYSCALL_CALL_RATE_CUM

The average number of system calls per second made to this function over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Due to the system call instrumentation on HP-UX, the fork and vfork system calls are double counted. In the case of fork and vfork, one process starts the system call, but two processes exit.

HP-UX lightweight system calls, such as umask, do not show up in the Glance System Calls display, but will get added to the global system call rates. If a process is being traced (debugged) using standard debugging tools (such as adb or xdb), all system calls used by that process will show up in the System Calls display while being traced.

SYSCALL_CPU_TOTAL_TIME

The CPU time, in seconds, during the interval spent executing this system calls. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

SYSCALL_CPU_TOTAL_TIME_CUM

The CPU time, in seconds, over the cumulative collection time spent executing this system calls.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

SYSCALL_INTERVAL

The amount of time in the interval.

SYSCALL_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TBL_BUFFER_CACHE_AVAIL

The size (in KBs unless otherwise specified) of the file systembuffer cache on the system.

On HP-UX 11i v2 and below, these buffers are used for all file system IO operations, as well as all other block IO operations in the system (exec, mount, inode reading, and some device drivers). If dynamic buffer cache is enabled, the system allocates a percentage of available memory not less than dbc_min_pct nor more than dbc_max_pct, depending on the system needs at any given time. On systems with a static buffer cache, this value will remain equal to bufpages, or not less than dbc_min_pct nor more than dbc_max_pct.

On HP-UX 11i v3 and above the limits of the file systembuffer cache which is still being used for file system metadata are automatically set to certain percentages of filecache_min and filecache_max.

On SUN, this value is obtained by multiplying the system page size times the number of buffer headers (nbuf). For example, on a SPARCstation 10 the buffer size is usually (200 (page size buffers) * 4096 (bytes/page) = 800 KB).

NOTE: (For SUN systems with VERITAS File System installed) Veritas implemented their Direct I/O feature in their file system to provide mechanism for bypassing the Unix system buffer cache while retaining the on disk structure of a file system. The way in which Direct I/O works involves the way the system buffer cache is handled by the Unix OS. Once the VERITAS file system returns with the requested block, instead of copying the content to a system buffer page, it copies the block into the application's buffer space. That's why if you have installed vxfs on your system, the TBL_BUFFER_CACHE_AVAIL can exceed the TBL_BUFFER_CACHE_HWM metric.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On AIX, this cache is used for all block IO.

On AIX System WPARs, this metric is NA.

TBL_BUFFER_CACHE_HIGH

The highest size (in KBs unless otherwise specified) of the buffer cache used in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_BUFFER_CACHE_MAX

On HP-UX 11i v2 and below, this metric represents the maximum size (in KBs unless otherwise specified) of the buffer cache. This corresponds to the kernel configuration parameter “dbc_max_pct”. On systems with a dynamic buffer cache, the cache does not exceed this limit. On systems with a fixed buffer cache, the cache size is equal to the value reported, which is based on the dbc_max_pct or bufpages settings.

On HP-UX 11i v3 and above, this metric represents the maximum size (in KBs unless otherwise specified) of the file cache. This corresponds to the kernel configuration parameter “filecache_max”.

TBL_BUFFER_CACHE_MIN

On HP-UX 11i v2 and below, this metric represents the minimum size (in KBs unless otherwise specified) of the buffer cache. This corresponds to the kernel configuration parameter “dbc_min_pct”. On systems with a dynamic buffer cache, the cache does not shrink below this limit. On systems with a fixed buffer cache, the cache size is equal to the value reported, which is based on the dbc_min_pct or bufpages settings.

On HP-UX 11i v3 and above, this metric represents the minimum size (in KBs unless otherwise specified) of the file cache. This corresponds to the kernel configuration parameter “filecache_min”.

TBL_BUFFER_CACHE_USED

The size (in KBs unless otherwise specified) of the sum of the currently used buffers.

On HP-UX 11i v2 and below, this is normally greater than the amount requested due to internal fragmentation of the buffer cache. Since this is a cache, it is normal for it to be filled. The buffer cache is used to stage all block IOs to disk. On a dynamic buffer cache configuration, this metric is always equal to TBL_BUFFER_CACHE_AVAIL. With dynamic buffer cache, the system allocates a percentage of available memory not less than dbc_min_pct nor more than dbc_max_pct, depending on the system needs at any given time. On systems with a static buffer cache, this value will remain equal to bufpages, or not less than dbc_min_pct nor more than dbc_max_pct. With a static buffer cache, this metric shows the amount of memory within the configured size that is actually used.

On HP-UX 11i v3 and above this metric value represents the usage of the file systembuffer cache which is still being used for file system metadata.

On AIX, this is normally greater than the amount requested due to internal fragmentation of the buffer cache. Since this is a cache, it is normal for it to be filled. The buffer cache is used to stage all block IOs to disk.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_BUFFER_HEADER_AVAIL

This is the maximum number of headers pointing to buffers in the file systembuffer cache.

On HP-UX, this is the configured number, not the maximum number. This can be set by the “nbuf” kernel configuration parameter. nbuf is used to determine the maximum total number of buffers on the system.

On HP-UX, these are used to manage the buffer cache, which is used for all block IO operations. When nbuf is zero, this value depends on the “bufpages” size of memory (see System Administration Tasks manual). A value of “na” indicates either a dynamic buffer cache configuration, or the nbuf kernel parameter has been left unconfigured and allowed to “float” with the bufpages parameter. This is not a maximum available value in a fixed buffer cache configuration. Instead, it is the initial configured value. The actual number of used buffer headers can grow beyond this initial value.

On SUN, this value is “nbuf”.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

TBL_BUFFER_HEADER_USED

The number of buffer headers currently in use.

On HP-UX, this dynamic value will rarely change once the system boots. During the system bootup, the kernel allocates a large number of buffer headers and the count is likely to stay at that value after the bootup completes. If the value increases beyond the initial boot value, it will not decrease. Buffer headers are allocated in kernel memory, not user memory, and therefore, will not decrease. This value can exceed the available or configured number of buffer headers in a fixed buffer cache configuration.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_BUFFER_HEADER_UTIL

The percentage of buffer headers currently used.

On HP-UX, a value of “na” indicates either a dynamic buffer cache configuration, or the nbuf kernel parameter has been left unconfigured and allowed to “float” with the bufpages parameter.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_BUFFER_HEADER_UTIL_HIGH

The highest percentage of buffer header used in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On HP-UX, a value of “na” indicates either a dynamic buffer cache configuration, or the nbuf kernel parameter has been left unconfigured and allowed to “float” with the bufpages parameter.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_DNLC_CACHE_AVAIL

The configured number of entries in the incore directory name cache.

On HP-UX, the directory name lookup cache is used to minimize sequential searches through directory entries for pathname components during pathname to inode translations. Such translations are done whenever a file is accessed through its filename. The cache holds the inode cache table offset for recently referenced pathname components. Pathname components that exceed 15 characters are not cached.

Any HP-UX system call that includes a path parameter can result in directory name lookup cache activity, including but not limited to system calls such as open, stat, exec, lstat, unlink. Each component of a path parameter is parsed and converted to an inode separately, therefore several dnlc hits per path are possible.

High directory name cache hit rates on HP-UX will be seen on systems where pathname component requests are frequently repeated. For example, when users or applications work in the same directory where they repeatedly list or open the same files, cache hit rates will be high.

Unusually low cache hit rates might be seen on HP-UX systems where users or applications access many different directories in no particular pattern. Low cache hit rates can also be an indicator of an underconfigured inode cache. When an inode cache is too small, the kernel will more frequently have to flush older inode cache and their corresponding directory name cache entries in order to make room for new inode cache entries.

On HP-UX, the directory name lookup cache is static in size and is allocated in kernel memory. As a result, it is not affected by user memory constraints. The size of the cache is stored in the kernel variable “ncsize” and is not directly tunable by the system administrator; however, it can be changed indirectly by tuning other tables used in the formula to compute the “ncsize”. The formula is:

```
ncsize = MAX(((nproc+16+maxusers)+
           32+(2*npty)),ninode)
```

Note that ncsize is always >= ninode which is the default size of the inode cache. This is because the directory name cache contains inode table offsets for each cached pathname component.

TBL_FILE_LOCK_AVAIL

The configured number of file or record locks that can be allocated on the system. Files and/or records are locked by calls to lockf(2). On Linux kernel versions 2.4 and above, available file orrecord locks is a dynamic value which can grow upto max unsigned long.

TBL_FILE_LOCK_USED

The number of file or record locks currently in use. One file can have multiple locks. Files and/or records are locked by calls to lockf(2).

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

TBL_FILE_LOCK_UTIL

The percentage of configured file or record locks currently in use. On Linux 2.4 and above kernel versions, this may not give correct picture as file or record locks available may change dynamically and can grow upto max unsigned long.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_LOCK_UTIL_HIGH

The highest percentage of configured file or record locks that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_TABLE_AVAIL

The number of entries in the file table.

On HP-UX and AIX, this is the configured maximum number of the file table entries used by the kernel to manage open file descriptors.

On HP-UX, this is the sum of the “nfile” and “file_pad” values used in kernel generation.

On SUN, this is the number of entries in the file cache. This is a size. All entries are not always in use. The cache size is dynamic. Entries in this cache are used to manage open file descriptors. They are reused as files are closed and new ones are opened. The size of the cache will go up or down in chunks as more or less space is required in the cache.

On AIX, the file table entries are dynamically allocated by the kernel if there is no entry available. These entries are allocated in chunks.

TBL_FILE_TABLE_USED

The number of entries in the file table currently used by file descriptors.

On SUN, this is the number of file cache entries currently used by file descriptors.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_TABLE_UTIL

The percentage of file table entries currently used by file descriptors.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_TABLE_UTIL_HIGH

The highest percentage of entries in the file table used by file descriptors in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_INODE_CACHE_AVAIL

On HP-UX, this is the configured total number of entries for the incore inode tables on the system. For HP-UX releases prior to 11.2x, this value reflects only the HFS inode table. For subsequent HP-UX releases, this value is the sum of inode tables for both HFS and VxFS file systems (ninode plus vxfs_ninode).

On HP-UX, file system directory activity is done through inodes that are stored on disk. The kernel keeps a memory cache of active and recently accessed inodes to reduce disk IOs. When a file is opened through a pathname, the kernel converts the pathname to an inode number and attempts to obtain the inode information from the cache based on the filesystem type. If the inode entry is not in the cache, the inode is read from disk into the inode cache.

On HP-UX, the number of used entries in the inode caches are usually at or near the capacity. This does not necessarily indicate that the configured sizes are too small because the tables may contain recently used inodes and inodes referenced by entries in the directory name lookup cache. When a new inode cache entry is required and a free entry does not exist, inactive entries referenced by the directory name cache are used. If after freeing inode entries only referenced by the directory name cache does not create enough free space, the message “inode: table is full” message may appear on the console. If this occurs, increase the size of the kernel parameter, ninode. Low directory name cache hit ratios may also indicate an underconfigured inode cache.

On HP-UX, the default formula for the ninode size is:

```
  ninode = ((nproc+16+maxusers)+32+
            (2*npty)+(4*num_clients))
```

On all other Unix systems, this is the number of entries in the inode cache. This is a size. All entries are not always in use. The cache size is dynamic.

Entries in this cache are reused as files are closed and new ones are opened. The size of the cache will go up or down in chunks as more or less space is required in the cache.

Inodes are used to store information about files within the file system. Every file has at least two inodes associated with it (one for the directory and one for the file itself). The information stored in an inode includes the owners, timestamps, size, and an array of indices used to translate logical block numbers to physical sector numbers. There is a separate inode maintained for every view of a file, so if two processes have the same file open, they both use the same directory inode, but separate inodes for the file.

TBL_INODE_CACHE_HIGH

On HP-UX and OSF1, this is the highest number of inodes that have been used in any one interval over the cumulative collection time.

On HP-UX, file system directory activity is done through inodes that are stored on disk. The kernel keeps a memory cache of active and recently accessed inodes to reduce disk IOs. When a file is opened through a pathname, the kernel converts the pathname to an inode number and attempts to obtain the inode information from the cache based on the filesystem type. If the inode entry is not in the cache, the inode is read from disk into the inode cache.

On HP-UX, the number of used entries in the inode caches are usually at or near the capacity. This does not necessarily indicate that the configured sizes are too small because the tables may contain recently used inodes and inodes referenced by entries in the directory name lookup cache. When a new inode cache entry is required and a free entry does not exist, inactive entries referenced by the directory name cache are used. If after freeing inode entries only referenced by the directory name cache does not create enough free space, the message “inode: table is full” message may appear on the console. If this occurs, increase the size of the kernel parameter, ninode. Low directory name cache hit ratios may also indicate an underconfigured inode cache.

On HP-UX, the default formula for the ninode size is:

```
ninode = ((nproc+16+maxusers)+32+
         (2*npty)+(4*num_clients))
```

On all other Unix systems, this is the largest size of the inode cache in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_INODE_CACHE_USED

The number of inode cache entries currently in use.

On HP-UX, this is the number of “non-free” inodes currently used. Since the inode table contains recently closed inodes as well as open inodes, the table often appears to be fully utilized. When a new entry is needed, one can usually be found by reusing one of the recently closed inode entries.

On HP-UX, file system directory activity is done through inodes that are stored on disk. The kernel keeps a memory cache of active and recently accessed inodes to reduce disk IOs. When a file is opened through a pathname, the kernel converts the pathname to an inode number and attempts to obtain the inode information from the cache based on the filesystem type. If the inode entry is not in the cache, the inode is read from disk into the inode cache.

On HP-UX, the number of used entries in the inode caches are usually at or near the capacity. This does not necessarily indicate that the configured sizes are too small because the tables may contain recently used inodes and inodes referenced by entries in the directory name lookup cache. When a new inode cache entry is required and a free entry does not exist, inactive entries referenced by the directory name cache are used. If after freeing inode entries only referenced by the directory name cache does not create enough free space, the message “inode: table is full” message may appear on the console. If this occurs, increase the size of the kernel parameter, ninode. Low directory name cache hit ratios may also indicate an underconfigured inode cache.

On HP-UX, the default formula for the ninode size is:

```
  ninode = ((nproc+16+maxusers)+32+
           (2*npty)+(4*num_clients))
```

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_BUFFER_AVAIL

The maximum achievable size (in KBs unless otherwise specified) of the message queue buffer pool on the system.

Each message queue can contain many buffers which are created whenever a program issues a msgsnd(2) call. Each of these buffers is allocated from this buffer pool.

Refer to the ipcs(1) man page for more information.

This value is determined by taking the product of the three kernel configuration variables “msgseg”, “msgssz” and “msgmni”. If the value adds up to a value > 2048GB, “o/f” may be reported on some platforms.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_MSG_BUFFER_HIGH

The largest size (in KBs unless otherwise specified) of the message queues in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_BUFFER_USED

The current total size (in KBs unless otherwise specified) of all IPC message buffers. These buffers are created by msgsnd(2) calls and released by msgrcv(2) calls.

On HP-UX and OSF1, this field corresponds to the CBYTES field of the “ipcs -qo” command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_TABLE_AVAIL

The configured maximum number of message queues that can be allocated on the system. A message queue is allocated by a program using the msgget(2) call.

Refer to the ipcs(1) man page for more information.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_MSG_TABLE_USED

On HP-UX, this is the number of message queues currently in use.

On all other Unix systems, this is the number of message queues that have been built.

A message queue is allocated by a program using the msgget(2) call. See ipcs(1) to list the message queues.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_TABLE_UTIL

The percentage of configured message queues currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_TABLE_UTIL_HIGH

The highest percentage of configured message queues that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_PROC_TABLE_AVAIL

The configured maximum number of the proc table entries used by the kernel to manage processes. This number includes both free and used entries.

On HP-UX, this is set by the NPROC value during system generation.

AIX has a “dynamic” proc table, which means that AVAIL has been set higher than should ever be needed.

On AIX System WPARs, this metric is NA.

TBL_PROC_TABLE_USED

The number of entries in the proc table currently used by processes.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_PROC_TABLE_UTIL

The percentage of proc table entries currently used by processes.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

TBL_PROC_TABLE_UTIL_HIGH

The highest percentage of entries in the proc table used by processes in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_PTY_AVAIL

The configured number of entries used by the pseudo-teletype driver on the system. This limits the number of pty logins possible.

For HP-UX, both telnet and rlogin use streams devices.

Note: On Solaris 8, by default, the number of ptys is unlimited but restricted by the size of RAM. If the number of ptys is unlimited, this metric is reported as “na”.

TBL_PTY_USED

The number of pseudo-teletype driver (pty) entries currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_PTY_UTIL

The percentage of configured pseudo-teletype driver (pty) entries currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_PTY_UTIL_HIGH

The highest percentage of configured pseudo-teletype driver (pty) entries in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SEM_TABLE_AVAIL

The configured number of semaphore identifiers (sets) that can be allocated on the system.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_SEM_TABLE_USED

On HP-UX, this is the number of semaphore identifiers currently in use.

On all other Unix systems, this is the number of semaphore identifiers that have been built.

A semaphore identifier is allocated by a program using the semget(2) call. See ipcs(1) to list semaphores.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SEM_TABLE_UTIL

The percentage of configured semaphores identifiers currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SEM_TABLE_UTIL_HIGH

The highest percentage of configured semaphore identifiers that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_ACTIVE

The size (in KBs unless otherwise specified) of the shared memory segments that have running processes attached to them. This may be less than the amount of shared memory used on the system because a shared memory segment may exist and not have any process attached to it.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_AVAIL

The maximum achievable size (in MB unless otherwise specified) of the shared memory pool on the system.

This is a theoretical maximum determined by multiplying the configured maximum number of shared memory entries (shmmni) by the maximum size of each shared memory segment (shmmax). Your system may not have enough virtual memory to actually reach this theoretical limit - one cannot allocate more shared memory than the available reserved space configured for virtual memory.

It should be noted that this value does not include any architectural limitations. (For example, on a 32-bit kernel, there is an addressing limit of 1.75 GB.). If the value adds up to a value > 2048TB, "o/f" may be reported on some platforms.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_SHMEM_REQUESTED

The size (in KBs unless otherwise specified) of the sum of the currently requested shared memory segments.

This may be more than shared memory used if any segments are swapped out. It also may be less than shared memory used due to internal fragmentation of the shared memory pool.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_TABLE_AVAIL

The configured number of shared memory segments that can be allocated on the system.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_SHMEM_TABLE_USED

On HP-UX, this is the number of shared memory segments currently in use.

On all other Unix systems, this is the number of shared memory segments that have been built. This includes shared memory segments with no processes attached to them.

A shared memory segment is allocated by a program using the shmget(2) call. Also refer to ipcs(1).

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_TABLE_UTIL

The percentage of configured shared memory segments currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_TABLE_UTIL_HIGH

The highest percentage of configured shared memory segments that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_USED

The size (in KBs unless otherwise specified) of the shared memory segments.

Additionally, it includes memory segments to which no processes are attached. If a shared memory segment has zero attachments, the space may not always be allocated in memory. See ipcs(1) to list shared memory segments.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TTBIN_TRANS_COUNT
TT_CLIENT_BIN_TRANS_COUNT

The number of completed transactions in this range during the last interval.

TTBIN_TRANS_COUNT_CUM
TT_CLIENT_BIN_TRANS_COUNT_CUM

The number of completed transactions in this range over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TTBIN_UPPER_RANGE

The upper range (transaction time) for this TT bin.

There are a maximum of nine user-defined transaction response time bins (TTBIN_UPPER_RANGE). The last bin, which is not specified in the transaction configuration file (ttdconf.mwc on Windows or ttd.conf on UNIX platforms), is the overflow bin and will always have a value of -2 (overflow). Note that the values specified in the transaction configuration file cannot exceed 2147483.6, which is the number of seconds in 24.85 days. If the user specifies any values greater than 2147483.6, the numbers reported for those bins or Service Level Objectives (SLO) will be -2.

TT_ABORT
TT_CLIENT_ABORT

The number of aborted transactions during the last interval for this transaction.

TT_ABORT_CUM
TT_CLIENT_ABORT_CUM

The number of aborted transactions over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_ABORT_WALL_TIME
TT_CLIENT_ABORT_WALL_TIME

The total time, in seconds, of all aborted transactions during the last interval for this transaction.

TT_ABORT_WALL_TIME_CUM
TT_CLIENT_ABORT_WALL_TIME_CUM

The total time, in seconds, of all aborted transactions over the cumulative collection time for this transaction class.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_APPNO

The registered ARM Application/User ID for this transaction class.

TT_APP_NAME

The registered ARM Application name.

TT_CACHE_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on CACHE during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CACHE_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on CACHE over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CDFS_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on CDFS (waiting in the CD-ROM driver for Compact Disc file system IO to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CDFS_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of transaction was blocked on CDFS (waiting in the CD-ROM driver for Compact Disc file system IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CLIENT_ADDRESS
TT_INSTANCE_CLIENT_ADDRESS

The correlator address. This is the address where the child transaction originated.

TT_CLIENT_ADDRESS_FORMAT
TT_INSTANCE_CLIENT_ADDRESS_FORMAT

The correlator address format. This shows the protocol family for the client network address. Refer to the ARM API Guide for the list and description of supported address formats.

TT_CLIENT_CORRELATOR_COUNT

The number of client or child transaction correlators this transaction has started over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_CLIENT_TRAN_ID
TT_INSTANCE_CLIENT_TRAN_ID

A numerical ID that uniquely identifies the transaction class in this correlator.

TT_COUNT
TT_CLIENT_COUNT

The number of completed transactions during the last interval for this transaction.

TT_COUNT_CUM
TT_CLIENT_COUNT_CUM

The number of completed transactions over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_CPU_CSWITCH_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction spent in context switching during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_CSWITCH_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction spent in context switching over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_INTERRUPT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction spent processing interrupts during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_INTERRUPT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction spent processing interrupts over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which

have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_NICE_TIME_PER_TRAN

The average time, in seconds, that each niced instance of the transaction was using the CPU in user mode during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_NICE_TIME_PER_TRAN_CUM

The average time, in seconds, that each niced instance of the transaction was in user mode over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_NNICE_TIME_PER_TRAN

The average time, in seconds, that each negatively niced instance of the transaction was using the CPU in user mode during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_NNICE_TIME_PER_TRAN_CUM

The average time, in seconds, that each negatively niced instance of the transaction was in user mode over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_NORMAL_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was in user mode at normal priority during the interval.

Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_NORMAL_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was in user mode at normal priority over the cumulative collection time. Normal priority user mode CPU excludes CPU used at real-time and nice priorities.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_REALTIME_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was in user mode at a realtime priority during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_REALTIME_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was in user mode at a realtime priority over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_SYSCALL_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was in system mode, excluding interrupt or context processing, during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_SYSCALL_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was in system mode, excluding interrupt or context processing, over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_SYS_MODE_TIME_PER_TRAN

The average CPU time in system mode in the context of each completed instance of the transaction during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_SYS_MODE_TIME_PER_TRAN_CUM

The average CPU time in system mode in the context of each completed instance of the transaction over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which

have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_TOTAL_TIME_PER_TRAN

The average total CPU time, in seconds, consumed by each completed instance of the transaction during the interval.

Total CPU time is the sum of the CPU time components for a process or kernel thread, including system, user, context switch, interrupt processing, realtime, and nice utilization values.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_TOTAL_TIME_PER_TRAN_CUM

The average total CPU time consumed by each completed instance of the transaction over the cumulative collection time. CPU time is in seconds unless otherwise specified.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_CPU_USER_MODE_TIME_PER_TRAN

The average time, in seconds, each completed instance of the transaction was using the CPU in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_CPU_USER_MODE_TIME_PER_TRAN_CUM

The average time, in seconds, each completed instance of the transaction was using the CPU in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_FS_READ_PER_TRAN

The average number of file system physical disk reads made by each completed instance of the transaction during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical reads generated by user file system access and do not include virtual memory reads, system reads (inode access), or reads relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical reads in this category. They appear under virtual memory reads.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_FS_READ_PER_TRAN_CUM

The average number of file system physical disk reads made by each completed instance of the transaction over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are physical reads generated by user file system access and do not include virtual memory reads, system reads (inode access), or reads relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical reads in this category. They appear under virtual memory reads.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_FS_WRITE_PER_TRAN

The average number of file system physical disk writes made by each completed instance of the transaction during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

These are physical writes generated by user file system access and do not include virtual memory writes, system writes (inode updates), or writes relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical writes in this category. They appear under virtual memory writes.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_FS_WRITE_PER_TRAN_CUM

The average number of file system physical disk writes made by each completed instance of the transaction over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

These are physical writes generated by user file system access and do not include virtual memory writes, system writes (inode updates), or writes relating to raw disk access. An exception is user files accessed via the mmap(2) call, which does not show their physical writes in this category. They appear under virtual memory writes.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which

have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_LOGL_IO_PER_TRAN

The average number of logical IOs made by (or for) each completed instance of the transaction during the interval. NFS mounted disks are not included in this list.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk).

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_LOGL_IO_PER_TRAN_CUM

The average number of logical IOs made by (or for) each completed instance of the transaction over the cumulative collection time. NFS mounted disks are not included in this list.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk).

On many Unix systems, logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_LOGL_READ_PER_TRAN

The average number of disk logical reads made by each completed instance of the transaction during the interval. Calls destined for NFS mounted files are not counted.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_LOGL_READ_PER_TRAN_CUM

The average number of disk logical reads made by each completed instance of the transaction over the cumulative collection time. Calls destined for NFS mounted files are not counted.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the read system calls that are directed to disk devices. Also counted are read system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which

have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_LOGL_WRITE_PER_TRAN

Average number of disk logical writes made by each completed instance of the transaction during the interval. Calls destined for NFS mounted files are not counted.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_LOGL_WRITE_PER_TRAN_CUM

Average number of disk logical writes made by each completed instance of the transaction over the cumulative collection time. Calls destined for NFS mounted files are not counted.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting "o/f" (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

On many Unix systems, logical disk IOs are measured by counting the write system calls that are directed to disk devices. Also counted are write system calls made indirectly through other system calls, including writev, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, send, sento, sendmsg, and ipcsend.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_PHYS_IO_PER_TRAN

The average number of physical disk IOs per second made by each completed instance of the transaction during the interval.

For transactions which run for less than the measurement interval, this metric is normalized over the measurement interval. For example, a transaction ran for 1 second and did 50 IOs during its life. If the measurement interval is 5 seconds, it is reported as having done 10 IOs per second. If the measurement interval is 60 seconds, it is reported as having done 50/60 or 0.83 IOs per second.

"Disk" in this instance refers to any locally attached physical disk drives (that is, "spindles") that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_PHYS_IO_PER_TRAN_CUM

The average number of physical disk IOs per second made by each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

For transactions which run for less than the measurement interval, this metric is normalized over the measurement interval. For example, a transaction ran for 1 second and did 50 IOs during its life. If the measurement interval is 5 seconds, it is reported as having done 10 IOs per second. If the measurement interval is 60 seconds, it is reported as having done 50/60 or 0.83 IOs per second.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_PHYS_READ_PER_TRAN

The average number of physical reads made by (or for) each completed instance of the transaction during the last interval.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_PHYS_READ_PER_TRAN_CUM

The average number of physical reads made by (or for) each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_PHYS_WRITE_PER_TRAN

The average number of physical writes made by (or for) each completed instance of the transaction during the last interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_PHYS_WRITE_PER_TRAN_CUM

The average number of physical writes made by (or for) each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all

completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_RAW_READ_PER_TRAN

The average number of raw reads made for each completed instance of the transaction during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_RAW_READ_PER_TRAN_CUM

The average number of raw reads made for each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_RAW_WRITE_PER_TRAN

The average number of raw writes made for each completed instance of the transaction during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_RAW_WRITE_PER_TRAN_CUM

The average number of raw writes made for each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_SYSTEM_READ_PER_TRAN

The average number of file system management physical disk reads made for each completed instance of the transaction during the interval.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_SYSTEM_READ_PER_TRAN_CUM

The average number of file system management physical disk reads made for each instance completed of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_SYSTEM_WRITE_PER_TRAN

The average number of file system management physical disk writes made for each completed instance of the transaction during the interval.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_SYSTEM_WRITE_PER_TRAN_CUM

The average number of file system management physical disk writes made for each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

File system management IOs are the physical accesses required to obtain or update internal information about the file system structure (inode access). Accesses or updates to user data are not included in this metric.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_VM_READ_PER_TRAN

The average number of virtual memory reads made for each completed instance of the transaction during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_VM_READ_PER_TRAN_CUM

The average number of virtual memory reads made for each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_VM_WRITE_PER_TRAN

The average number of virtual memory writes made for each completed instance of the transaction during the interval.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_VM_WRITE_PER_TRAN_CUM

The average number of virtual memory writes made for each completed instance of the transaction over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_DISK_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on DISK during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_DISK_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on DISK over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_FAILED
TT_CLIENT_FAILED

The number of Failed transactions during the last interval for this transaction name.

TT_FAILED_CUM
TT_CLIENT_FAILED_CUM

The number of failed transactions over the cumulative collection time for this transaction name.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_FAILED_WALL_TIME
TT_CLIENT_FAILED_WALL_TIME

The total time, in seconds, of all failed transactions during the last interval for this transaction name.

TT_FAILED_WALL_TIME_CUM
TT_CLIENT_FAILED_WALL_TIME_CUM

The total time, in seconds, of all failed transactions over the cumulative collection time for this transaction name.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_GOLDENRESOURCE_INTERVAL

The amount of time in the collection interval.

TT_GOLDENRESOURCE_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_GRAPHICS_WAIT_TIME_PER_TRAN

The average time that each completed instance of the transaction was blocked on graphics (waiting for their graphics operations to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_GRAPHICS_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on graphics (waiting for their graphics operations to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all

completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_INFO

The registered ARM Transaction Information for this transaction.

TT_INODE_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked onINODE (waiting for an inode to be updated or to become available) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_INODE_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked onINODE (waiting for an inode to be updated or to become available) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_INPROGRESS_COUNT

The number of transactions in progress (started, but not stopped) at the end of the interval for this transaction class.

TT_INSTANCE_ID

A numerical ID that uniquely identifies this transaction instance at the end of the interval.

TT_INSTANCE_PROC_ID

The ID of the process that started or last updated the transaction instance.

TT_INSTANCE_START_TIME

The time this transaction instance started.

TT_INSTANCE_STOP_TIME

The time this transaction instance stopped. If the transaction instance is currently active, the value returned will be -1. It will be shown as “na” in Glance and GPM to indicate that the transaction instance did not stop during the interval.

TT_INSTANCE_THREAD_ID

The ID of the kernel thread that started or last updated the transaction instance.

TT_INSTANCE_UPDATE_COUNT

The number of times this transaction instance called update since the start of this transaction instance.

TT_INSTANCE_UPDATE_TIME

The time this transaction instance last called update. If the transaction instance is currently active, the value returned will be -1. It will be shown as “na” in Glance and GPM to indicate that a call to update did not occur during the interval.

TT_INSTANCE_WALL_TIME

The elapsed time since this transaction instance was started.

TT_INTERVAL
TT_CLIENT_INTERVAL

The amount of time in the collection interval.

TT_INTERVAL_CUM
TT_CLIENT_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_IPC_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked onIPC during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_IPC_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked onIPC over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to

report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_JOBCTL_WAIT_TIME_PER_TRAN

The average time that each completed instance of the transaction was blocked on job control (having been stopped with the job control facilities) during the interval. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_JOBCTL_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on job control (having been stopped with the job control facilities) over the cumulative collection time. Job control waits include waiting at a debug breakpoint, as well as being blocked attempting to write (from background) to a terminal which has the “stty tostop” option set.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_LAN_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on LAN (waiting for IO over the LAN to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_LAN_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on LAN (waiting for IO over the LAN to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all

completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_MEASUREMENT_COUNT

The number of user defined measurements for this transaction class.

TT_MEM_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on memory during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_MEM_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on memory over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance

agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_MSG_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on messages (waiting for message queue operations to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_MSG_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on messages (waiting for message queue operations to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_NAME

The registered transaction name for this transaction.

TT_NFS_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on NFS (waiting for its network file system IO to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_NFS_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on NFS (waiting for its network file system IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_OTHER_IO_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on “other IO” during the interval. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_OTHER_IO_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on “other IO” over the cumulative collection time. “Other IO” includes all IO directed at a device (connected to the local computer) which is not a terminal or LAN. Examples of “other IO” devices are local printers, tapes, instruments, and disks. Time waiting for character (raw) IO to disks is included in this measurement. Time waiting for file systembuffered IO to disks will typically been seen as IO or CACHE wait. Time waiting for IO to NFS disks is reported as NFS wait.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_OTHER_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on other (unknown) activities during the interval. This includes transactions that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_OTHER_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on other (unknown) activities over the cumulative collection time. This includes transactions that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_PIPE_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked onPIPE (waiting for pipe communication to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_PIPE_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked onPIPE (waiting for pipe communication to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_PRI_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on priority during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_PRI_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on priority over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the

system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_RESOURCE_INTERVAL

The amount of time in the collection interval.

TT_RESOURCE_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_RPC_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on RPC (waiting for its remote procedure calls to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_RPC_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on RPC (waiting for its remote procedure calls to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which

have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_SEM_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked onsemaphores (waiting on a semaphore operation to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_SEM_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked onsemaphores (waiting on a semaphore operation to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is

older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_SLEEP_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A transaction enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_SLEEP_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on SLEEP (waiting to awaken from sleep system calls) over the cumulative collection time. A transaction enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_SLO_COUNT
TT_CLIENT_SLO_COUNT

The number of completed transactions that violated the defined Service Level Objective (SLO) by exceeding the SLO threshold time during the interval.

TT_SLO_COUNT_CUM
TT_CLIENT_SLO_COUNT_CUM

The number of completed transactions that violated the defined Service Level Objective by exceeding the SLO threshold time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_SLO_PERCENT

The percentage of transactions which violate service level objectives.

TT_SLO_THRESHOLD

The upper range (transaction time) of the Service Level Objective (SLO) threshold value. This value is used to count the number of transactions that exceed this user-supplied transaction time value.

TT_SOCKET_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on sockets (waiting for its IO to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_SOCKET_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on sockets (waiting for its IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all

completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_STREAM_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on streams IO (waiting for a streams IO operation to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_STREAM_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on streams IO (waiting for a streams IO operation to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_SYS_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on system blocked on SYSTM (that is, system resources) during the interval. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_SYS_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on SYSTM (that is, system resources) over the cumulative collection time. These resources include data structures from the LVM, VFS, UFS, JFS, and Disk Quota subsystems. “SYSTM” is the “catch-all” wait state for blocks on system resources that are not common enough or long enough to warrant their own stop state.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_TERM_IO_WAIT_TIME_PER_TRAN

The average time, in seconds, that each completed instance of the transaction was blocked on terminal IO (waiting for its terminal IO to complete) during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_TERM_IO_WAIT_TIME_PER_TRAN_CUM

The average time, in seconds, that each completed instance of the transaction was blocked on terminal IO (waiting for its terminal IO to complete) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_TOTAL_WAIT_TIME_PER_TRAN

The average total time that each completed instance of the transaction spent blocked during the interval.

Per-transaction performance resource metrics represent an average for all completed instances of the given transaction during the interval.

If there are no completed transaction instances during an interval, then there are no resources accounted, even though there may be in-progress transactions using resources which have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance during an interval, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances during an interval for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances during the interval. To obtain the total accumulated resource consumption for all completed transactions during an interval, multiply the resource metric by the number of completed transaction instances during the interval (TT_COUNT).

TT_TOTAL_WAIT_TIME_PER_TRAN_CUM

The average total time that each completed instance of the transaction spent blocked during over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

Cumulative per-transaction performance resource metrics represent an average for all completed instances of the given transaction over the cumulative collection time.

If there are no completed transaction instances over the cumulative collection time, then there are no resources accounted, even though there may be in-progress transactions using resources which

have not completed. Resource metrics for in-progress transactions will be shown in the interval after they complete (that is, after the process has called arm_stop).

If there is only one completed transaction instance over the cumulative collection time, then the resources attributed to the transaction will represent the resources used by the process between its call to arm_start and arm_stop, even if arm_start was called before the current interval. Thus, the resource usage time or wall time per transaction can exceed the current collection interval time.

If there are several completed transaction instances over the cumulative collection time for a given transaction, then the resources attributed to the transaction will represent an average for all completed instances over the cumulative collection time. To obtain the total accumulated resource consumption for all completed transactions over the cumulative collection time, multiply the resource metric by the number of completed transaction instances over the cumulative collection time (TT_COUNT_CUM).

TT_TRAN_1_MIN_RATE

For this transaction name, the number of completed transactions calculated to a 1 minute rate. For example, if you completed five of these transactions in a 5 minute window, the rate is one transaction per minute.

TT_TRAN_ID

The registered ARM Transaction ID for this transaction class as returned by arm_getid(). A unique transaction id is returned for a unique application id (returned by arm_init), tran name, and meta data buffer contents.

TT_UNAME

The registered ARM Transaction User Name for this transaction.

If the arm_init function has NULL for the appl_user_id field, then the user name is blank. Otherwise, if “*” was specified, then the user name is displayed.

For example, to show the user name for the armsample1 program, use:

```
appl_id = arm_init(“armsample1”,“*”,0,0,0);
```

To ignore the user name for the armsample1 program, use:

```
appl_id = arm_init(“armsample1”,NULL,0,0,0);
```

TT_UPDATE
TT_CLIENT_UPDATE

The number of updates during the last interval for this transaction class. This count includes update calls for completed and in progress transactions.

TT_UPDATE_CUM
TT_CLIENT_UPDATE_CUM

The number of updates over the cumulative collection time for this transaction class. This count includes update calls for completed and in progress transactions.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_USER_MEASUREMENT_AVG
TT_INSTANCE_USER_MEASUREMENT_AVG
TT_CLIENT_USER_MEASUREMENT_AVG

If the measurement type is a numeric or a string, this metric returns “na”.

If the measurement type is a counter, this metric returns the average counter differences of the transaction or transaction instance during the last interval. The counter value is the difference observed from a counter between the start and the stop (or last update) of a transaction.

If the measurement type is a gauge, this returns the average of the values passed on any ARM call for the transaction or transaction instance during the last interval.

TT_USER_MEASUREMENT_COUNT
TT_INSTANCE_USER_MEASUREMENT_COUNT
TT_CLIENT_USER_MEASUREMENT_COUNT

This returns the total number of times the associated user defined metric (UDM) was sampled during the last interval.

TT_USER_MEASUREMENT_MAX
TT_INSTANCE_USER_MEASUREMENT_MAX
TT_CLIENT_USER_MEASUREMENT_MAX

If the measurement type is a numeric or a string, this metric returns “na”.

If the measurement type is a counter, this metric returns the highest measured counter value over the life of the transaction or transaction instance. The counter value is the difference observed from a counter between the start and the stop (or last update) of a transaction.

If the measurement type is a gauge, this metric returns the highest value passed on any ARM call over the life of the transaction or transaction instance.

TT_USER_MEASUREMENT_MIN
TT_INSTANCE_USER_MEASUREMENT_MIN
TT_CLIENT_USER_MEASUREMENT_MIN

If the measurement type is a numeric or a string, this metric returns “na”.

If the measurement type is a counter, this metric returns the lowest measured counter value over the life of the transaction or transaction instance. The counter value is the difference observed from a counter between the start and the stop (or last update) of a transaction.

If the measurement type is a gauge, this metric returns the lowest value passed on any ARM call over the life of the transaction or transaction instance.

TT_USER_MEASUREMENT_NAME
TT_INSTANCE_USER_MEASUREMENT_NAME
TT_CLIENT_USER_MEASUREMENT_NAME

The name of the user defined transactional measurement. The length of the string complies with the ARM 2.0 standard, which is 44 characters long (there are 43 usable characters since this is a NULL terminated character string).

TT_USER_MEASUREMENT_STRING1024_VALUE
TT_INSTANCE_USER_MEASUREMENT_STRING1024_VALUE
TT_CLIENT_USER_MEASUREMENT_STRING1024_VALUE

The last value of the user defined measurement of type string 1024. This type is not implemented and the value is always “na”.

TT_USER_MEASUREMENT_STRING32_VALUE
TT_INSTANCE_USER_MEASUREMENT_STRING32_VALUE
TT_CLIENT_USER_MEASUREMENT_STRING32_VALUE

The last value of the user defined measurement of type string 32.

TT_USER_MEASUREMENT_TYPE
TT_INSTANCE_USER_MEASUREMENT_TYPE
TT_CLIENT_USER_MEASUREMENT_TYPE

The type of the user defined transactional measurement.

```
 1 = ARM_COUNTER32
 2 = ARM_COUNTER64
 3 = ARM_CNTRDIVR32
 4 = ARM_GAUGE32
 5 = ARM_GAUGE64
 6 = ARM_GAUGEDIVR32
 7 = ARM_NUMERICID32
 8 = ARM_NUMERICID64
 9 = ARM_STRING8    (max 8 chars)
10 = ARM_STRING32   (max 32 chars)
11 = ARM_STRING1024 (max 1024 char -- not implemented)
```

TT_USER_MEASUREMENT_VALUE
TT_INSTANCE_USER_MEASUREMENT_VALUE
TT_CLIENT_USER_MEASUREMENT_VALUE

The last value of the user defined measurement of type counter, gauge, numeric ID, or string 8. Both 32 and 64 bit numeric types are returned as 64 bit values.

TT_WALL_TIME
TT_CLIENT_WALL_TIME

The total time, in seconds, of all transactions completed during the last interval for this transaction.

TT_WALL_TIME_CUM
TT_CLIENT_WALL_TIME_CUM

The total time, in seconds, of all transactions completed over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.

TT_WALL_TIME_PER_TRAN
TT_CLIENT_WALL_TIME_PER_TRAN

The average transaction time, in seconds, during the last interval for this transaction.

TT_WALL_TIME_PER_TRAN_CUM
TT_CLIENT_WALL_TIME_PER_TRAN_CUM

The average transaction time, in seconds, over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or thread) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to Glance, if available for the given platform), whichever occurred last.

On HP-UX, all cumulative collection times and intervals start when the midaemon starts. On other Unix systems, non-process collection time starts from the start of the performance tool, process collection time starts from the start time of the process or measurement start time, which ever is older. Regardless of the process start time, application cumulative intervals start from the time the performance tool is started.

On systems where the performance components are 32-bit or where the 64-bit model is LLP64 (Windows), all INTERVAL_CUM metrics will start reporting “o/f” (overflow) after the performance agent (or the midaemon on HPUX) has been up for 466 days and the cumulative metrics will fail to report accurate data after 497 days. On Linux, Solaris and AIX, if measurement is started after the system has been up for more than 466 days, cumulative process CPU data won't include times accumulated prior to the performance tool's start and a message will be logged to indicate this.