HP Operations Manager for UNIX

for the HP Integrity (Itanium) operating system

Software Version: 8.21/8.30

Performance Guide

Manufacturing Part Number: none

Document Release Date: March 2009

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 Notices

Trademark Notices

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Intel®, Intel Inside®, Intel Inside logo, Itanium®, and Pentium® are trademarks of Intel Corporation in the U.S. and other countries.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

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

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

Pentium® is a U.S. registered trademark of Intel Corporation.

UNIX® is a registered trademark of the Open Group.

Documentation Updates

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

- Software Version number, which indicates the software version.
  - The number before the period identifies the major release number.
  - The first number after the period identifies the minor release number.
  - The second number after the period represents the minor-minor release number.
- Document Release Date, which changes each time the document is updated.

To check for recent updates or to verify that you are using the most recent edition, visit the following URL:

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

You can visit the HP Software support web site at:

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 software 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 HP Software 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 find more information about access levels, go to:

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

To register for an HP Passport ID, go to:

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

Contents

Introduction

This document shows you how to optimize the performance of your HP Operations Manager (HPOM) for UNIX environment. It provides performance tips for fine-tuning an existing HPOM for UNIX environment or for setting up a new HPOM for UNIX environment.

Audience

This document is intended for the person who sets up and maintains the HPOM for UNIX environment.

Prerequisites

Before reading this guide, make sure that you have in-depth knowledge of HPOM for UNIX, and are very familiar with its architecture, concepts, and acronyms.

Disclaimer

The results listed in this guide are valid only for the following versions of HPOM for UNIX:

- **First Performance Test Run** (March 2006)
  - HPOM for UNIX 8.21
  - HP-UX 11i v2 (HP-UX 11.23 (Itanium))
  - Oracle 10.1.0.4/64
- **Second Performance Test Run** (August 2008)
  - HPOM for UNIX 8.30
  - HP-UX 11i v3 (HP-UX 11.31 (Itanium))
  - Oracle 10.2.0.3/64

Chapters Summary

This guide is organized as follows:

Performance Considerations

This chapter describes the critical factors that affect application performance in an HPOM for UNIX environment. It summarizes high-level parameters, application-related parameters, and agent node management.

Performance Findings

This chapter describes the trade-offs between performance, security, and costs in an HPOM for UNIX environment. It summarizes measured and unmeasured performance findings, management server configuration variables, HPOM for UNIX 8.10 (PA-RISC) and 8.21 (Itanium) performance, management server performance, Java GUI performance, and HPOM for UNIX 8.30 (IA64) performance.

Test Environment

This chapter describes the environment used in performance tests of HPOM for UNIX. It summarizes the configuration of management servers, display stations, LoadRunner servers, managed nodes, and message generation.

Test Results

This chapter describes the results of performance tests of HPOM for UNIX. It includes detailed results of message throughput tests, Java GUI startup time tests, and history message download time tests.

Appendix A. Tuning the VXFS File System

This appendix explains how parameters for the database file system were tuned, based on recommendation from performance specialists for HP-UX 11.23.

Appendix B. Memory and Kernel Parameter Formulas

This appendix lists the formulas for the memory and kernel parameter requirements that `ovoinstall` uses for your information (as of the 8.31 `ovoinstall` package from October 2008).

Related Documents

Although this guide provides an overview of HPOM server and HPOM Java GUI performance, it does not attempt to cover all aspects and scenarios of HPOM for UNIX performance.

For parameters you can use to fine-tune HPOM server performance, see *HP Operations Manager for UNIX Server Configuration Variables* manual.

For parameters you can use to fine-tune HPOM agents, see the *HP Operations Manager HTTPS Agent Configuration Variables* manual.

For issues related to HPOM HTTPS agents, see the corresponding chapters in the *HP Operations for UNIX 8.10 Performance Guide* (March 2005).

For the latest versions of the HPOM for UNIX and HPOM agent manuals, check the following web site periodically:

http://support.openview.hp.com/selfsolve/manuals

Product: Operations Manager for UNIX

1 Performance Considerations

HP Operations Manager is a key component of the HP Business Service Management (BSM) solution. BSM maps business services to their underlying applications, infrastructure, and network components. This mapping helps you to analyze the business impact of IT problems and to reduce the potential costs of IT service downtime.

With the growing importance of products addressing BSM, HP Operations Manager (HPOM) for UNIX is used increasingly in large computing environments. HPOM for UNIX is the tool of choice for regional, enterprise, and worldwide business management.

To set up a robust and scalable HPOM environment, it is critical to evaluate architectural and performance considerations. The biggest challenge when measuring the performance of distributed client-server applications, such as HPOM for UNIX, is not the measurement itself. The biggest challenge is determining the set of parameters that have an effect on application performance. Because it is not always possible to provide general rules, this guide focuses on critical areas and limitations that affect application performance.

High-Level Parameters

From a high-level point of view, there are three different kinds of parameters that affect application performance:

- **Static parameters**

  Parameters related to the HP Operations management server (HPOM server), the HP Operations agent (HPOM agent), or both (for example, disk space requirements, operating system kernel configuration, and HPOM configuration).

- **Dynamic parameters**

  Parameters related to the HPOM server, HPOM agent, the HPOM user interface, and the Oracle Database (for example, memory usage of corresponding processes, CPU utilization, disk input/output, and so on).

- **Network parameters**

  Parameters related to network bandwidth, network utilization, and protocol overhead.

Application-Related Parameters

In addition, there are other application-related parameters:

- **Users**

  Number of HPOM operators working in parallel.

- **Messages**

  Number of HPOM messages per second received by the management server. Type and complexity of messages to be processed by the management server (for example, messages forwarded to the trouble ticket system, messages copied or diverted to the Message Stream Interface, duplicate message counting, and so on).

- **Message browser**

  Number of active and history messages held in the message browser.

- **Nodes**

  Number of nodes in the topology or object database maintained and monitored by HPOM.

- **Service objects**

  Number of service objects in Service Navigator.

- **Custom message attributes**

  Number of custom message attributes (CMAs) in HPOM messages.

Size of Managed Environment

It is not easy to determine how many nodes can be managed by a single HPOM server. There is no fixed limit for the number of nodes that can be controlled by an HPOM server. The capability of an HP Operations management server is based on the number and type of messages to be processed. As a result, the number of nodes that can be managed by a server depends largely on how the management domain is set up. For example, using more HPOM agent features (for example, local automatic actions and message filtering) improves the performance of the management server. There are customer environments in which 2,000 or more HPOM agents report to one HPOM server.

For large-scale environments, it is recommended that you set up HPOM Flexible Management (also known as "Manager-of-Manager" or "MoM") to spread the workload. Such a setup helps ensure a highly available solution that includes disaster recovery scenarios. For details, see the *High Availability Through HPOM Server Pooling* white paper.

For parameters you can use to fine-tune HPOM agents, see the *HP Operations Manager HTTPS Agent Configuration Variables* manual.

For issues related to HPOM HTTPS agents, see the corresponding chapters in the *HP Operations for UNIX 8.10 Performance Guide* (March 2005).

2 Performance Findings

In large-scale HP Operations Manager (HPOM) for UNIX environments, many factors determine the optimal setup with the best performance. In the long run, there are trade-offs between performance, security, hardware and software costs, and the total cost of ownership.

This document highlights a few key performance findings:

- Comparison of HPOM for UNIX on HP Integrity (Itanium) (versions 8.21 and 8.30) and HPOM for UNIX on HP PA-RISC (version 8.10)
- Various scenarios of the HPOM message processing and throughput on the HPOM server
- Startup time for the HP Operations Java UI (including Service Navigator)
- Repetition of selected test cases with HPOM for UNIX 8.30, as well as a few new test cases (for example, message forwarding from management server to management server, message storm recovery, and so on)

It is difficult to compare the performance results for HPOM for UNIX 8.21 and 8.30, because different hardware setups, different operating system versions, different Oracle Database versions, and different load generators have been applied.

The HPOM server on HP Integrity (IA64) is a 32-bit application, even though its operating system and Oracle Database are 64-bit.

Summary of Performance Findings

The following HPOM for UNIX performance findings have been measured:

- **Management server**
  - The number of used CPUs heavily depends on the number of concurrent Java UIs, since each Java UI (with a different user login) is served by its own instance of an opcuiwww process.
  - Depending on the hardware, number, and usage profile of concurrent GUIs and MSI applications, the HPOM server can sustainably process several hundred HPOM messages per second.
  - Although the number of managed nodes has no significant impact on both the Java GUI and Oracle Database message throughput, it does have an impact on some HPOM server processes.
  - If HPOM message duplicate suppression is enabled, message keys should be explicitly set for the messages. Performance of duplicate HPOM message processing is greatly improved by the auto key feature (QXCR1000424253), which is introduced with the HPOM for UNIX 8.31 server patch. The OPC_CREATE_AUTO_KEYS variable is enabled by default.
  - Using MSI on the HPOM agent and the HPOM server with more than five simultaneous MSI programs results in decreasing message throughput.
  - Using the configuration variable `OPCMSGM_USE_GUI_THREAD = TRUE` increases the message processing rate of the HPOM server.
  - HPOM servers can survive massive event storms, and then continue normal operations after a short period of time.

- When message bursts are not avoidable, the time of silence (that is, very low new incoming message volume) is normally long enough to process all pending messages.

- **Manager-of-Manager (MoM)**
  - In the MoM setup, forwarding only those messages required on the destination HPOM server minimizes performance overhead.
  - In the MoM setup, the time to synchronize HPOM message acknowledgements is linear and based on the number of acknowledged messages.
- **Oracle Database**
  - Reducing the bulk of acknowledged messages to less than 5,000 messages avoids disk and network bottlenecks.
  - HPOM history message download speed is about 400 to 500 messages per second.
  - Time required to download history messages depends on the configuration of the Oracle Database parameters, particularly the log file groups and buffers.
  - When downloading a large number of HPOM history messages, using a higher bulk commit size than the default improves the performance.
- **Motif GUI**

  The number of managed nodes in the HPOM for UNIX user's realm has a significant impact on the startup time of the Motif GUI.
- **Java GUI (including Service Navigator)**
  - As more Java GUIs are started, message throughput decreases in a linear fashion, as long as there are still CPUs available to serve the associated GUI processes. If there are no longer any available CPUs, the message throughput decreases faster.
  - HTTPS is the recommended communication method to the Java GUI. It is faster and more secure than HTTP.
  - Number of managed nodes has no significant impact on the start-up time of the Java GUI.
  - A significant impact on the start-up time of the Java GUI is observed with 50,000+ active messages to be loaded.
  - Keeping the number of active messages as low as possible (by using duplicate suppression, event correlation, and so on) reduces the start-up time for the Java GUI.
  - Java GUI preference "Show All Messages" should be used only if really needed.
  - Only the number of Java GUIs needed at the moment should be started and running.
  - When using CMAs, the additional performance degradation of the Java UI start-up time and the message throughput is low. It is recommended that you keep the number of CMAs at or below 10 CMAs per message.
  - To increase the usability of the Java GUI, it is recommended that acknowledged messages should be marked only (by setting the HPOM server configuration variable `OPC_DIRECT_ACKN_LIMIT` to `0`). Using this option will ensure that the acknowledged messages get moved to the History table automatically at a later point of time.
  - If deep Service Navigator graphs are used (for example, service graphs with more than five levels), it is recommended that you use the feature "Service Load on Demand" to avoid long delays in the start-up time of the Java GUI.

- **HTTPS agent**
  - Depending on the hardware and other competing applications, one HTTPS agent is able to send more than 500 messages per second (using the HPOM agent `opcmsg()` C API).
  - Maximum transfer rate from the agents to the server is faster than the message processing rate on the server.

Unmeasured Performance Findings

The following HPOM for UNIX performance findings have *not* been measured:

- **HTTPS agent performance**

  You can find some HPOM HTTPS agent performance-related measures in the *HP Operations Manager for UNIX 8.10 Performance and Configuration Guide*.

- **Local or remote Oracle Database**

  Running a remote Oracle Database offloads the HPOM server with CPU and main memory usage. However, remote SQL statements may also have performance and security-related limitations.

- **Server Pooling**

  HPOM for UNIX Server Pooling provides an easy-to-administer method for managing large-scale environments combined with the capability to quickly switch the workload to different HPOM servers if an HPOM server system is not reachable or is in maintenance mode. For details, see the *High Availability Through HPOM Server Pooling* white paper. HPOM for UNIX Server Pooling is also an attractive alternative to hardware-based high-availability setups.

- **Physical system or virtual machine**

  HPOM for UNIX 8 on HP-UX can be run on a physical system or on a HP-UX virtual machine (VM). Although running HPOM for UNIX on a VM can decrease performance somewhat, the ease of administration (for example, setting up multiple HPOM servers using VM images, separating "competing" applications in different VMs, and so on) make it an option worth considering.

- **Speed of name resolution on the management server**

  During the startup of the HPOM server, all node names currently recognized by HPOM for UNIX are resolved to fill the corresponding HPOM server process caches for fast processing.

  Depending on the speed of Domain Name Service (DNS), this name resolution could take a considerable amount of time.

Management Server Configuration Variables

Table 1 lists a subset of HPOM server configuration variables that influence overall HPOM for UNIX performance. The list includes only those configuration variables that either have a different default or that have been introduced with HPOM for UNIX 8 server patches.

The list is *not* comprehensive. Other important HPOM server configuration variables settings could potentially have a negative impact on overall HPOM server performance.

Table 1 Key Configuration Variables for the Management Server

Parameter	Recommended value (default)
OPCMSGM_USE_GUI_THREAD	TRUE (FALSE)
OPC_DIRECT_ACKN_LIMIT	0 (50)
OPC_UPDWN_COMMIT_COUNT	1000 (100)

Setting Configuration Variables

To set configuration variables for the HPOM server, use the following CLI call:

```
# ovconfchg -ovrg server -ns opc –set <variable> <value>
```

For details, see the *HP Operations Manager for UNIX Server Configuration Variables* manual.

HPOM for UNIX 8.10 (PA-RISC) and 8.21 (Itanium)

This section compares the performance of HPOM for UNIX 8.10 (PA-RISC) and HPOM for UNIX 8.21 (Itanium).

Management Server Performance

For HPOM for UNIX 8.10 (PA-RISC) and 8.21 (Itanium), tests of message throughput show the following maximum message processing rates on HPOM servers:

- HPOM for UNIX 8.10 on PA-RISC processes up to 120 messages per second.
- HPOM for UNIX 8.21 on Itanium processes up to 333 messages per second.
- With HPOM for UNIX 8.21 IA64 (Itanium), message throughput is increased. For technical reasons, the test system was more powerful than the system used in the previous tests. However, the most significant impact is seen by enabling new configuration variables – such as OPCMSGM_USE_GUI_THREAD.
- When HPOM for UNIX 8.21 IA64 and PA32 (PA-RISC) were compared on nearly identical systems, no significant differences were observed.
- Some general performance improvements in HPOM for UNIX 8.21 are also available for the corresponding HPOM for UNIX 8 server patches for PA-RISC and Sun Solaris.

Java GUI Performance

The HPOM for UNIX 8.21 Java GUI starts faster than the HPOM for UNIX 8.1 Java GUI.

This difference is increasingly obvious under the following conditions:

- More active messages exist.
- Complex service graphs are assigned to the operators.

In addition, the start-up times for Service Navigator have been improved.

HPOM for UNIX 8.30 (IA64)

For HPOM for UNIX 8.30 (IA64), tests show the following results:

- Management server is able to withstand high load and volume usually with low resource consumption.
- Message throughput tests show a maximum message processing rate on the server of up to 226 messages per second.
- Java GUI is able to handle high load and volume capacity with a short start-up time.
- HPOM server is able to survive massive event storms, and then to continue normal operations after a short period of time.
- Configuring the Java GUI to use HTTPS-based communication improves performance.

3 Test Environment

This chapter describes the environment used in performance tests of HP Operations Manager (HPOM) for UNIX.

Management Server

This section describes the management server configurations used in performance tests of HPOM for UNIX.

Hardware and Software

Table 1 lists the hardware and software configuration used to test the HPOM for UNIX 8.21 management server.

Table 1 HPOM for UNIX 8.21 Management Server Configuration

Type	Configuration
Network	Dedicated 1 Gb network for the display stations and managed nodes
Machine type	HP RX8620/16
Processor	16 x 1.5 GHz
Main memory	128 GB [1]
Disk space	Internal disks for the operating system Disk array (striped) through a fiber channel (EVA 4000)
Operating system	HP-UX 11iv2 (11.23)
Swap space	512 MB [2]
Operating system patches	QPKBASE B.11.23.0512.034 QPKAPPS B.11.23.0512.034

1. This huge amount of memory was not explicitly requested for the tests and was not fully used during the test.

2. Only a small swap space was reserved because the system was equipped with a significant amount of main memory, and because the system was configured in such a way that swapping was performed entirely in memory. According to performance specialists, performance test results are less significant if systems swap on disks.

Table 2 lists the hardware and software configurations used to test the HPOM for UNIX 8.30 management server in medium, large, and MOM deployments.

Table 2 HPOM for UNIX 8.30 Management Server Configuration

	Deployments		
Type	**Medium**	**Large**	**MoM**
Network	Dedicated 1 Gb network for the display stations and managed nodes		
Machine type	HP RX8640		
Processor	8x1.6 GHz / 24 MB Montecito	14x1.6 GHz / 24 MB Montecito	Server 1 – 8x1.6 GHz / 24 MB Montecito Server 2 – 8x1.6 GHz / 24 MB Montecito
Main memory	64 GB	112 GB	Server 1 – 64 GB Server 2 – 64 GB
Operating system	HPUX11i-DC-OE B.11.31.0803		
Swap space	Only OS bare minimum configured, since all operations could be done in main memory. Swap space was configured for 512MB.		
Operating system patches	QPKBASE B.11.31.0803.318a SwMgmtMin B.11.31.0803.318 HWEnable11i B.11.31.0803.318a FEATURE11i B.11.31.0803.318b		

Kernel Configuration

This section describes the kernel configurations used in performance tests of HPOM for UNIX.

HPOM for UNIX 8.21 Parameters

During the HPOM for UNIX 8.21 setup, the kernel configuration was checked according to the parameters listed in Table 3.

Table 3 HPOM for UNIX 8.21 Configuration Parameters

Configuration Parameter	Value
Number of Java GUIs	50
Service Navigator GUIs	5
Number of Motif GUIs	10
Number of HTTPS agents	100
Number of DCE agents	10

Based on `ovoinstall` recommendations, the kernel parameters for the HPOM for UNIX 8.21 setup changed, as shown in Table 4.

Table 4 HPOM for UNIX 8.21 Kernel Parameters

Kernel Parameter	Old Value	New Value
msgmap	1026	4098
semume	64	250
shmseg	300	400
vps_ceiling	16	64

HPOM for UNIX 8.30 Parameters

Before the tests of HPOM for UNIX 8.30 began, some kernel parameters were adjusted based on the *HP Operations Manager for UNIX Installation Guide* (ovo.info.HP-UX.B.11.31.txt). The new kernel parameters are listed in Table 5.

Table 5 HPOM for UNIX 8.30 Kernel Parameters

Kernel Parameter	Value
msgmap	5026
maxdsiz	0x80000000 (2 GB)
maxfiles	4096
maxfiles_lim	4096
maxssiz	0x08000000 (128 MB)
maxssiz_64	0x800000000 (2 GB)
maxuprc	4995
nflocks	5071
ninode	42048
semmni	10000
semmns	0xf0000000
shmmax	4096
shmseg	512
nproc	5000

Hostname Name Resolution

To minimize the potential impact by a domain name service (DNS) for the hostname resolution, a local /etc/hosts file has been used.

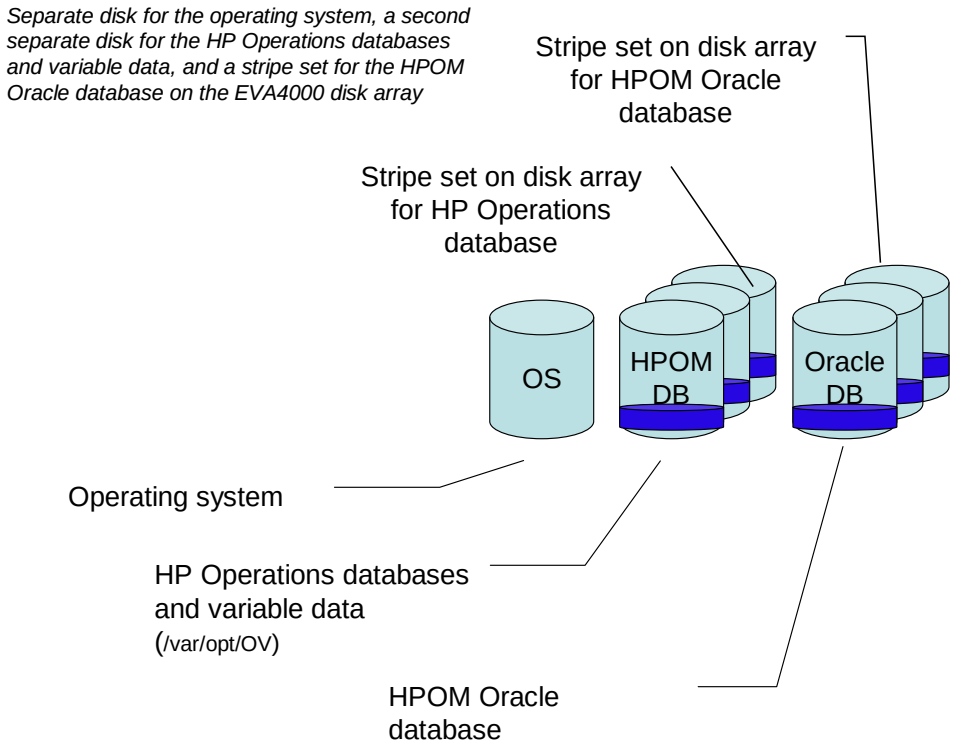
Disk Setup

This section describes the disk setup used in performance tests of HPOM for UNIX.

HPOM for UNIX 8.21

For all tests of HPOM for UNIX 8.21, the disk setup was fixed, based on the results of previous HPOM for UNIX performance tests. The fixed disk setup is shown in Figure 1.

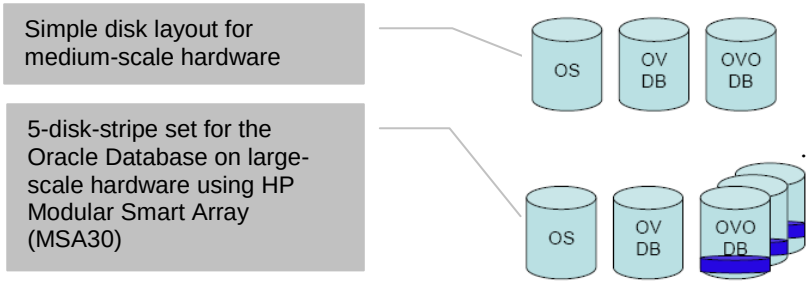
Figure 1 HPOM for UNIX 8.21 Fixed Disk Setup



HPOM for UNIX 8.30

For tests of HPOM for UNIX 8.30, the setup of the disks was varied, based on the hardware scale, as shown in Figure 2.

Figure 2 HPOM for UNIX 8.30 Varied Disk Setup



All disks used in the performance tests had the following specifications:

- 300 GB16 MB cache
- Ultra320 SCSI 80-pin interface
- Up to 125 MB per second sustained transfer rate

Based on recommendation from the HP European Performance Center, some of the `vxfs` file system parameters were tuned in `/etc/vx/tunefstab`. For details, see "Tuning the VXFS File System" on page 72.

HPOM for UNIX Installations

This section describes the HPOM for UNIX installations used in the performance tests.

HPOM for UNIX 8.21

The tests of HPOM for UNIX 8.21 used the following installation:

- Base Version A.08.20
- Server Patch PHSS_34111 (IA64 consolidated server A.08.21)

During Phase II of the HPOM for UNIX database configuration, the following error message displayed:

```
cannot allocation 4096 bytes of shared memory
```

The error message displayed because the Oracle parameter `sga_target` was set to 128 MB in the `initopenview.ora` file created by `ovdbsetupo1_opc.sh`. This value is too small, even for the creation of the HPOM database. It is recommended that you set it to at least 350 MB.

This problem is fixed in server version A.08.21 with patch PHSS_34111.

HPOM for UNIX 8.30

The tests of HPOM for UNIX 8.30 used the following installation:

- Base Version HPOM A.08.20
- HPOM patches:
  - PHSS_36632 – IA-64 Certificate Server A.08.25
  - PHSS_37678 – A.08.51 IA Agent Patch
  - PHSS_37679 – A.08.51 Windows Agent Patch
  - PHSS_38200 – IA-64 Consolidated Server A.08.30
  - PHSS_38202 – IA-64 Java GUI Client A.08.30
  - PHSS_38204 – IA-64 Java Configuration API A.08.30
  - Hotfix for message suppress duplicates processing (QXCR1000424253)
- HPOM for UNIX Configuration Value Pack (CVP) 3.1.2 (Midas Configurator) for easy HPOM for UNIX administration. No CVP-related performance tests were conducted.

RDBMS Installation

Performance tests of HPOM for UNIX used the following RDBMS installation:

- Oracle 10.1.0.4 / 64 bit for HPOM for UNIX 8.21 and Oracle 10.2.0.3 / 64 bit for HPOM for UNIX 8.30.
- Configurable parameters in file `initopenview.ora` [default values in square brackets]

Table 6 lists the Oracle parameters for HPOM for UNIX 8.21 and 8.30.

Table 6 HPOM for UNIX 8.21 and 8.30 Oracle Parameters

	Value [Default Value]	
Parameter	**HPOM for UNIX 8.21**	**HPOM for UNIX 8.30**
`sga_target`	512M [128M] Further tests have shown that a value of 350 MB is acceptable. This value is now used in the patch A.08.21.	512MB [350MB]
`db_files`	80 [50]	
`db_file_multiblock_read_count`	32 [8]	32 [16]
Processes	200 [50]	500 [200]
`log_buffer`	1572864 [65536]	2 MB [65536]
Redo logs	5 x 100 MB [3 x 20 MB] 5 log groups with one log file each	
Buffer cache	1056 MB	

Display Stations and LoadRunner Servers

This section describes the Windows Java GUI display stations and LoadRunner servers used in performance tests of HPOM for UNIX.

HPOM for UNIX 8.21 Java GUI

Table 7 lists the Windows Java GUI displays stations used in performance tests of HPOM for UNIX 8.21.

Table 7 HPOM for UNIX 8.21 Java GUI

Machine Type	2 x DL580/4/3.3	1 x DL580/4/3.3
Main Memory	16 GB	13.7 GB
CPU	4 x 3.3 GHz Xeon	4 x 3.3 GHz Xeon
OS	Windows 2003	Windows 2003

HPOM for UNIX 8.30 Java GUI

Table 8 lists the Windows Java GUI displays stations and LoadRunner servers used in performance tests of HPOM for UNIX 8.30.

Table 8 HPOM for UNIX 8.30 Java GUI

Machine Type	8 x HP BL460c	1 x HP BL460c
Main Memory	6 GB	6 GB
CPU	2.83 GHz Quadcore Xeon	2.66 GHz Dualcore Xeon
OS	Windows 2003	Windows 2003

Managed Nodes

This section describes the managed nodes used in performance tests of HPOM for UNIX.

HPOM for UNIX 8.21

For performance tests of HPOM for UNIX 8.21, the real managed nodes (the nodes used for message generation) were added manually to the Node Bank, downloaded, and always reloaded with each of the test configurations.

Table 9 HPOM for UNIX Managed Nodes

	HP9000 Superdome	
Machine Type	**SD64A Part-1**	**SD64A Part-2**
Main memory	64 GB	64 GB
CPU	4 x PA-8800 1GHz	4 x PA-8800 1GHz
Operating system	HP-UX 11.11	HP-UX 11.11
Agent type	HTTPS	HTTPS

HPOM for UNIX 8.30

For performance tests of HPOM for UNIX 8.30, HP LoadRunner was used to simulate managed node, as there was no need for a real node.

For details, see "HPOM for UNIX 8.30 Agent Simulation and Message Generation" chapter below.

Message Generation

This section describes the message generation used in performance tests of HPOM for UNIX.

Managed Node

When working with managed nodes, you should be aware of the following:

- **Proxied nodes**

  A node is called a proxied node when the message is targeted for a different node than the actual physical node. This targeting is performed by specifying a different node name in the node parameter of the `opcmsg` API. Proxied nodes can be of network type IP or non-IP. Because it is not possible to have a real agent on a non-IP node, non-IP nodes are always proxied nodes.

- **Non-IP nodes**

  All nodes in the node groups are defined as ordinary HP Operations managed nodes of the machine type non-IP/`OTHER`, with a name, but without an IP address. Because these nodes are not reachable by using IP, they are called non-IP nodes.

- **Real managed nodes**

  The HPOM agent is installed on the real managed nodes. In most of the tests, the real managed nodes did not generate messages for themselves, but were used to generate the messages for the proxied nodes only.

- **Hostname resolution**

  The process `opcmsgi` tries to resolve the hostname for nodes given with the node parameter of `opcmsg`. Since dummy nodes were used, the normal DNS could not be used. Depending on the `/etc/nsswitch.conf` and the size of the `/etc/hosts` (if searched), the generation for nodes, especially of the type `Non IP` can take a very long time (timeouts of name servers and time used to search the `/etc/hosts`).

  To avoid these delays, a very short `/etc/hosts` file was introduced that contained only the necessary names. An `/etc/nsswitch.conf` file was constructed to direct the search to the files only.

- **Message throughput**

  Test results show differences in the message throughput of real nodes and proxied nodes. Based on these test results, a better performance in message throughput may be expected for real nodes.

Message Types

The Message Generator can create different types of messages, as described in Table 10. These types are flagged with a special code in the message text that triggers a dedicated condition in the assigned Message Interface template.

Table 10 Message Types Generated by the Message Generator

Message Type	Description
NO	Normal message.
OL	Normal message, but with the flag `on server log only`.
AA	Message with an automatic action (Echo50).
AN	Similar to message type AA, except the output of the automatic action is recorded as an annotation. (The size of annotation for the Echo50action is 50 characters.)
AC	Similar to message type AA, except the message is acknowledged automatically.
ACN	Similar to message type AC, except the output of the automatic action is recorded as an annotation.
NT	Message that is forwarded to the notification service.
TT	Message that is forwarded to the trouble ticket service.
IN	Message with fixed instructions.
MIX	Message mix comprised of the following: • 40% normal messages • 20% messages with fixed instructions • 20% messages that are forwarded to the notification service • 20% messages with automatic actions

HPOM for UNIX 8.21 Message Generation

For the performance tests of HPOM for UNIX 8.21, the Message Generator was used to create the test messages. The Message Generator consists of programs (using the `opcmsg` C API) and a Message Interface template.

The managed node used to generate the messages was not the critical factor in message throughput. In all scenarios tested, the management server itself was the element limiting the message throughput.

The Message Generator does the following:

- Generates messages for a configurable set of nodes (specified through the range of node numbers) and node groups.
- Generates a fixed number of messages per node, and then stops.
- Generates messages for only one message group per managed node.

You can start the Message Generator on real managed nodes running HP-UX.

You can configure the Message Generator by using a specification file that does the following:

- Assigns the real managed nodes to node groups specified on the command line.
- Maps the canonical node names `nodeng_num` used in message generator to real node names. (Canonical or systematic node names are used to specify the range of nodes for which messages are generated.)

HPOM for UNIX 8.30 Agent Simulation and Message Generation

For the performance tests of HPOM for UNIX 8.30, agents were simulated by HP LoadRunner. HP LoadRunner simulates an agent connection that behaves very much like a real agent connection. As such, the connection is treated as a real agent.

Each simulated agent creates a SSL connection. The simulated agent sends a message using a simple HTTPS call, exactly as a real agent does. HP LoadRunner is capable of simulating any message type by defining the specific XML structure of the relevant HTTPS call.

During the tests, the following message types were used:

- **Normal message**

  Normal (`NO`) message with a message text of about 30 characters

- **Normal message with CMAs**

  Normal (`NO`) messages with a CMA name length of 20 characters and CMA value length of 50 characters

4 Test Results

This chapter describes the results of performance tests of HP Operations Manager (HPOM) for UNIX. For a description of the test environment, see "Test Environment" on page 16.

The performance tests covered the following areas:

- **Message Throughput**

  How many messages the system could process:

  — Test: Varying the Number of Active Java GUIs on page 28

  — Test: Suppressing Message Duplicates on page 33

  — Test: Diverting Messages to MSI Programs on page 36

  — Test: Detailed Inspection of Message Processing on page 40

  — Test: Message Storm on page 43

  — Test: MoM Message Forwarding on page 46

- **Java GUI Start-Up Time**

  How long it took for the Java GUI to start up fully:

  — Test: Varying the Number of Managed Nodes and Messages on page 48

  — Test: Varying the Number of CMAs on page 53

  — Test: Message Acknowledgment on page 57

- **Service Navigator Start-Up Time**

  How long it took for Service Navigator to start up fully:

  — Test: Varying the Shape and Size of the Service Tree on page 61

- **History Download Performance**

  How quickly history messages could be removed:

  — Test: Downloading 100,000 History Messages on page 65

- **Message Throughput of IP vs. non-IP Nodes**

  How many messages the system could process on real nodes:

  — Test: Message Throughput on IP vs. non-IP Nodes on page 69

Message Throughput

Message throughput tests measured how many messages the system was able to process.

Test: Varying the Number of Active Java GUIs

This test varied the number of active Java GUIs on HPOM for UNIX 8.21 and 8.30.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Number of messages per second (message throughput) until all messages were shown in all Java GUIs
  - Number of messages per second (message throughput) until all messages were stored in the HPOM for UNIX Oracle Database
- **Parameters**
  - Number of Java GUIs
  - Number of managed nodes

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time until the last expected message was shown in all message browsers was the result of the test.
  - Time on the server to receive all messages (Receive Time of message) divided by the number of messages was the rate of messages stored in the HPOM Oracle Database.
- **Message Generator**
  - *HPOM for UNIX 8.21 only*
    - 5,000 messages with a severity of normal were used.
    - Node Bank had 2,000 managed nodes.
    - Messages were targeted to 100 nodes in 4 node groups (that is, 50 messages per node)
  - *HPOM for UNIX 8.30 only*
    - 10,000 messages with a severity of normal were used.
    - Node Bank had 2,000 and 4,000 managed nodes.
    - Messages were targeted to 100 nodes in 4 node groups (that is, 100 messages per node)
  - *HPOM for UNIX 8.21 and 8.30*
    - No history messages were generated.
    - Operators were responsible for 400 nodes.
- **Java GUI options**

- Different HPOM for UNIX user accounts were used.
- All messages were shown.
- Refresh interval was 5 seconds.[1]

- **HPOM server options**
  - `OPCMSGM_USE_GUI_THREAD` [2] = TRUE [3] [4]

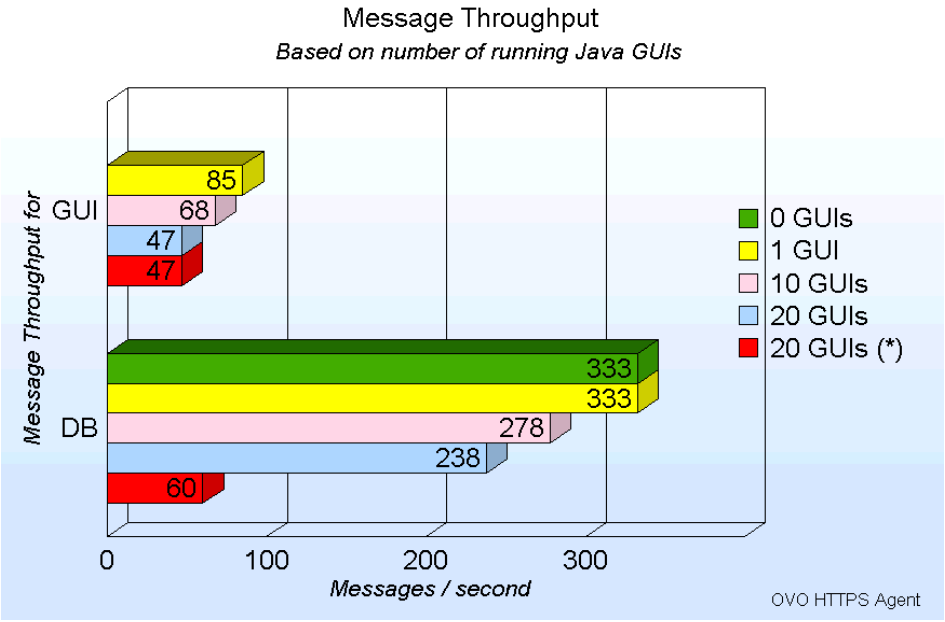
Results

This section describes the results of the varying the number of Java GUIs.

HPOM for UNIX 8.21

Figure 3 shows the message throughput for HPOM for UNIX 8.21, based on the number of running Java GUIs. As communication protocol HTTP has been used.
The message generator is an HPOM HTTPS agent. An additional test was performed with the HPOM server variable `OPCMSGM_USE_GUI_THREAD` not set. The result is shown as "20 GUIs (*)".

Figure 3 Message Throughput – Number of Java GUIs (HTTPS Agent)



[1] The refresh interval had an impact on the outcome of the bulk transfer (set via OPCUIWWW_BULK_MODE) of messages to the Java GUI. The smaller the refresh interval, the smaller the performance gain of the bulk transfer. For most of the Java GUI tests, the smaller refresh value was used to get more timely results. The recommendation for production use is still 30 seconds. This refresh value is the default in the Java GUI.

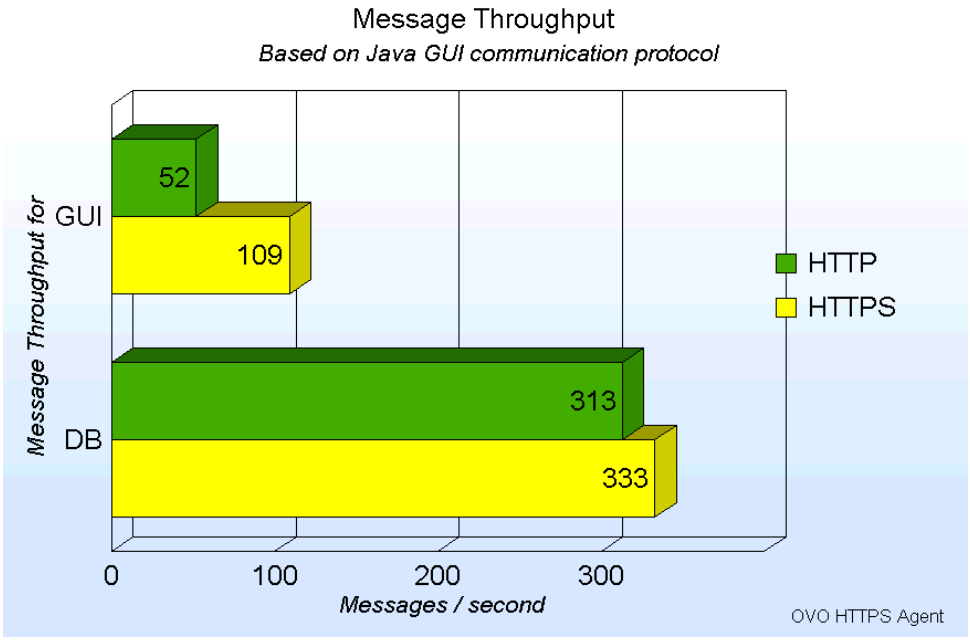
[2] This HPOM server configuration variable enables the HPOM message manager process to serve the HPOM for UNIX GUI using a dedicated thread, thereby decoupling the sending information to the display and handling database storage.

[3] HPOM server options were set with the ovconfchg –ovrg server –ns opc –set variable value.

[4] Since HPOM for UNIX 8.22, a newer value, NO_RPC, is available for OPCMSGM_USE_GUI_THREAD that enables further decoupling by changing the opcmsgm – opcuiwww process communication from DCE to queue files. Using NO_RPC is a prerequisite for using OPCUIWWW_NEW_MSG_NO_DB = TRUE (also new with HPOM for UNIX 8.22). In addition, HPOM for UNIX 8.22 adds the configuration parameter: OPCUIWWW_BULK_MODE = TRUE.
There have been no explicit tests performed using these new configuration parameters, but some further performance improvements are likely.

Figure 4 shows the impact of the network communication protocol on the Java GUI. That is, it contrasts HTTP with HTTPS communication. Two Java GUIs were used in this test.

Figure 4 Message Throughput – Communication Protocol



HPOM for UNIX 8.30

Figures 5 and 6 show message throughput for HPOM for UNIX 8.30, based on the number of running Java GUIs. The Java GUIs were configured using HTTPS-based communication and OPCMSGM_USE_GUI_THREAD was set to TRUE.
Figure 5 shows 2,000 nodes in the Node Bank.

Figure 5 Message Throughput – Number of Java GUIs (2,000 Nodes)

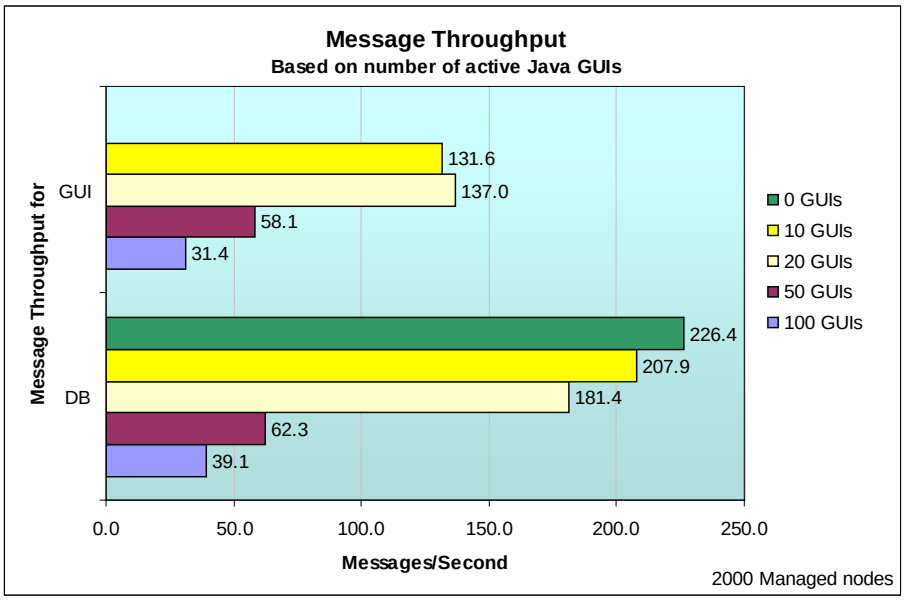
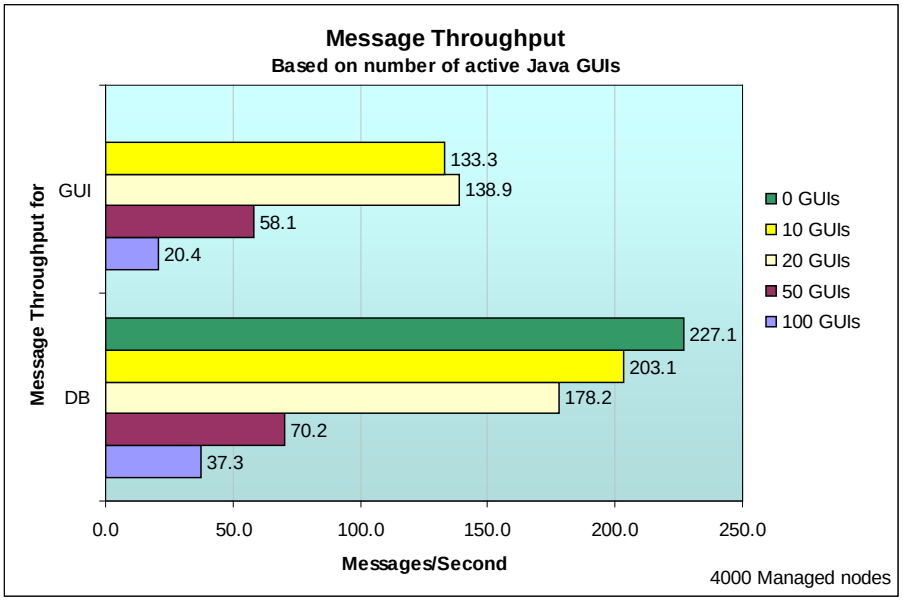


Figure 6 shows 4,000 nodes in the Node Bank.

Figure 6 Message Throughput – Number of Java GUIs (4,000 Nodes)



The test “Message Throughput – Communication Protocol” has not been repeated for HPOM for UNIX 8.30, since same conclusions can be drawn, that HTTPS communication protocol is the preferred one.

Interpretation

Based on the test results, the following facts can be derived:

- As more Java GUIs are started, the message throughput decreases in a linear fashion, as long as CPUs are still available to serve the associated GUI processes. If there are no longer any available CPUs, the message throughput decreases faster.
- Loading 100 GUIs when there are no messages arriving has a very low impact on the CPU (consumption is around 5%). Additional tests show that 200 or more GUIs can be loaded with no impact on the CPU.
- Message throughput in the form of messages arriving in the Oracle Database is higher than the message throughput in the form of message displaying in the Java message browser.
- Communicating with the Java GUI by using HTTPS doubles the message throughput in the form of messages displaying in the message browser, even though encryption and authentication are added.
- Only measured for HPOM of UNIX 8.21: The rate at which the messages are stored in the Oracle Database is slightly higher with HTTPS communication than with HTTP communication.
- Number of managed nodes has no significant impact on GUI or database message throughput

Conclusions

Based on the test results, the following conclusions may be drawn:

- Only the number of Java GUIs needed at the moment should be started. As more Java GUIs are started, the Oracle Database message throughput decreases.
- HTTPS should be used as the communication type for the Java GUI. This communication protocol is faster because of implicit data compression. The lower data volume more than compensates for the overhead of SSL communication.

Test: Suppressing Message Duplicates

This test suppressed message duplicates on HPOM for UNIX 8.21 and 8.30.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Number of messages per second (message throughput) until all messages were shown in the Motif Admin GUI
  - Number of messages per second (message throughput) until all messages were stored in the Oracle Database
- **Parameters**
  - Message duplicate counter active or inactive
  - Number of different message keys (0, 1)

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time until the last expected message was shown in the Motif Admin GUI message browser.
  - Number of messages stored in the Oracle Database divided by the time elapsed (receive time) was measured as the message throughput.
- **Message Generator**
  - 10,000 unique messages with a severity of normal (that is, the message text contained a unique number) were used.
  - All messages were available in the `msgmgrq` queue at the start of the test (that is, no messages arrived through the test).
  - Messages were targeted to 100 nodes in 4 node groups (that is, 100 messages per node).
  - Operators were responsible for 400 nodes.
  - Node Bank contained 2.000 managed nodes.
  - No history messages were used.
- **Java GUI options**
  - No Java GUIs were started.
- **Miscellaneous**
  - “Add duplications as annotations” feature was switched off.
  - HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.
  - *HPOM for UNIX 8.30 only:* New auto key feature (QXCR1000424253) was used.

In the previous performance tests, modifying the number of message keys showed very little impact on the computed metric values. As a result, for the current tests, it was decided to use only zero and one different message key.

Results

This section describes the results of suppressing message duplicates.

HPOM for UNIX 8.21

Table 12 lists the message throughput in messages per second on HPOM for UNIX 8.21.

Table 12 HPOM for UNIX 8.21 Message Throughput – Duplicates Counter

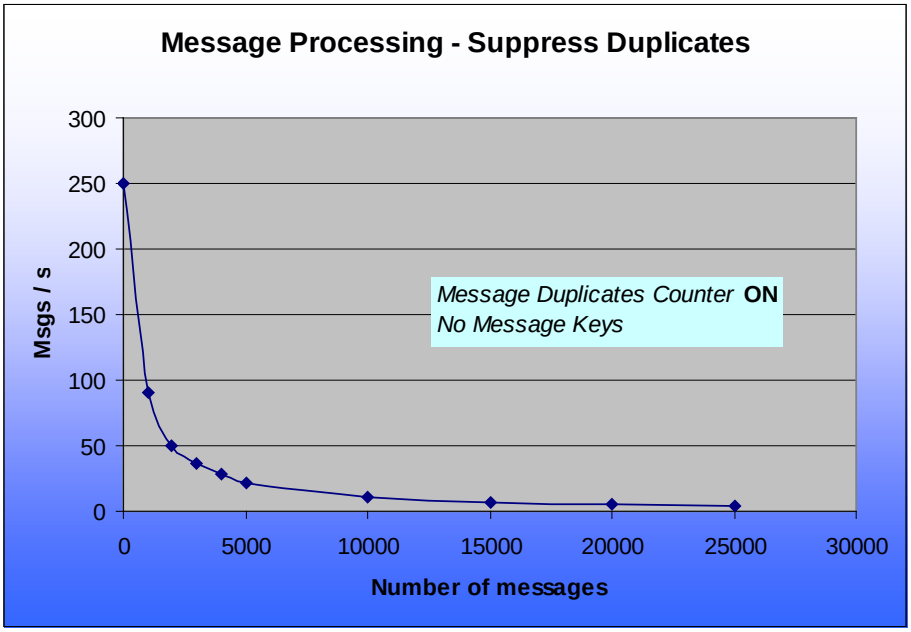
Message Keys	OFF	0	1
Msgs/sec	476	28	294
Messages in database	10,000	10,000	1

Legend

OFF	Message Duplicates Counter was switched off.
0	Message Duplicates Counter was switched on. No message keys were used.
1	Message Duplicates Counter was switched on. The same key was used for all messages

In a special test, the message processing speed was taken as a function of the number of messages in the active message browser, when duplicate suppression was enabled and no message keys were defined. As shown in Figure 7, message processing became progressively slower the more messages that were processed. These tests were conducted on a PA-RISC system configured like the IA64 system used for the other tests.

Figure 7 HPOM for UNIX 8.21 Message Throughput – Duplicates Counter



HPOM for UNIX 8.30

Table 13 lists the message throughput in messages per second and the number of messages stored in the database on HPOM for UNIX 8.30.

Table 13 HPOM for UNIX 8.30 Message Throughput – Duplicates Counter

Message Keys	OFF	0	1
Msgs/sec	226.4	137	263.2
Messages in database	10,000	10,000	1

Legend

OFF Message Duplicates Counter was switched off.

0 Message Duplicates Counter was switched on. No message keys were used.

1 Message Duplicates Counter was switched on. The same key was used for all messages.

Interpretation

Based on the test results, the following facts can be derived:

- If duplicates are to be counted without using message keys, the processing time is significantly increased.
- Number of different message keys has a small impact on performance. (That is, the more message keys you use, the longer it takes to process the messages.)
- If all messages have the same key, more messages are processed per second when the duplicates counter is switched on. Only one message is stored in the database and then updated (message counter field). If the duplicates counter is switched off, each new message must be inserted into the database.
- Lowest message throughput is measured when the duplicates counter is switched on and no message keys are used. Each message is compared with each existing active message in the database while the total number of messages increases.

Conclusions

Based on the test results, the following conclusions can be drawn:

- If message duplicate suppression is enabled, message keys should be attached to the messages—even if every message gets its own key.
- Performance of duplicate processing is greatly improved by the auto key feature (QXCR1000424253), which was officially introduced with the HPOM for UNIX 8.31 server patch, but already applied as hotfix for this performance test based on HPOM for UNIX 8.30.

Test: Sending Messages to MSI Programs

This test sent messages to Message Stream Interface (MSI) programs on HPOM for UNIX 8.21 and 8.30.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Number of messages per second (message throughput) until all messages arrived in the Motif Admin GUI from the MSI
- **Parameters**
  - Number of MSI programs registered in parallel
  - MSI active on the agent or server

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time until the last expected message was shown in the Motif Admin GUI message browser ("GUI Throughput")
  - Time on the server to receive all messages (Receive Time of message) divided by number of messages was measured as the "DB Throughput".
- **Message Generator**
  - 5,000 messages with a severity of normal were used.
  - Messages were targeted to 100 nodes in 4 node groups (that is, with 50 messages per node).
  - Operators were responsible for 400 nodes.
  - Node Bank had 2,000 managed nodes.
  - No history messages were used.
- **HPOM for UNIX server options**
  - HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.
- **Java GUI options**
  - No Java GUIs were started.
- **Miscellaneous**
  - *HPOM for UNIX 8.21 only*
    - All messages were diverted to one or more MSI programs using the agent or server MSI. The MSI programs simply wrote the unmodified message back to the message stream.
    - Because all MSI programs received the same set of messages, and each MSI program wrote its messages back to the MSI stream, a multiple of the initial 5,000 messages was seen in the message browser (5,000 messages per MSI program).
  - *HPOM for UNIX 8.30 only:*

All messages were copied to one or more MSI programs using the server MSI. The MSI programs did **not** return any message to HPOM for UNIX –meaning only the original messages were stored in the Oracle database.

Results

This section describes the results of diverting messages to MSI programs.

HPOM for UNIX 8.21

Figures 8 and 9 show the results of diverting messages to agent and server MSI programs on HPOM for UNIX 8.21.

Legend:

- **MSI Progs: None**

  No MSI program was active. Messages were processed directly without diverting, although forwarding to the MSI was switched on for the server and agent, as well as in the policy.

- **MSI Progs: Agent n x MSI**

  n MSI programs were active using the agent MSI. *n* times 5,000 messages were actually stored in the Oracle Database.

- **MSI Progs: Server n x MSI**

  n MSI programs were active using the server MSI. n times 5,000 messages were actually stored in the Oracle Database.

Figure 8 shows the message throughput when diverting messages to agent MSI (HTTPS agent) programs on HPOM for UNIX 8.21.

Figure 8 HPOM for UNIX 8.21 Message Throughput – Agent MSI

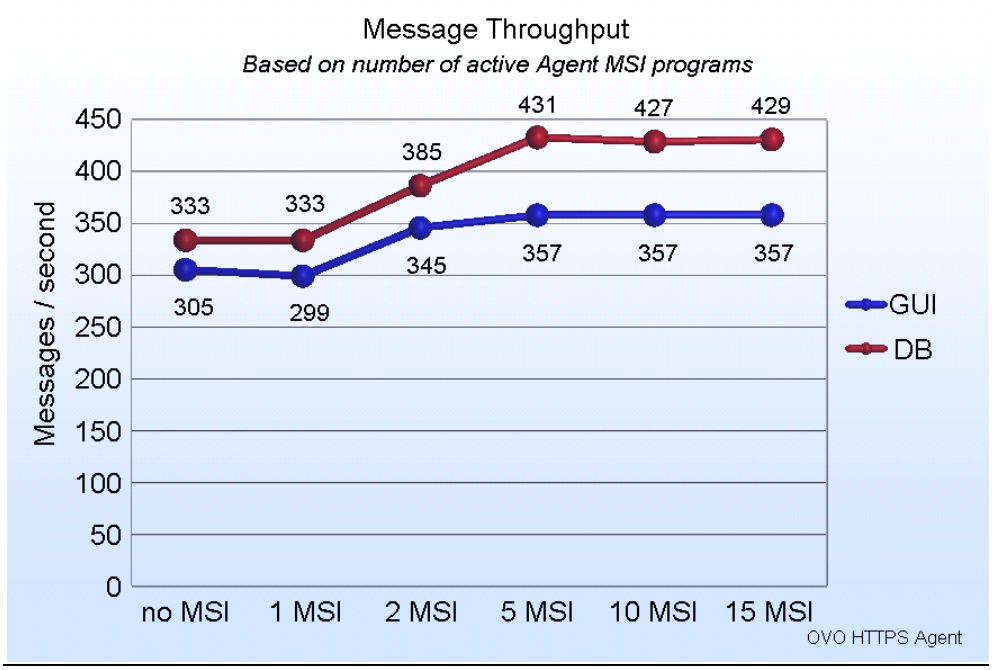
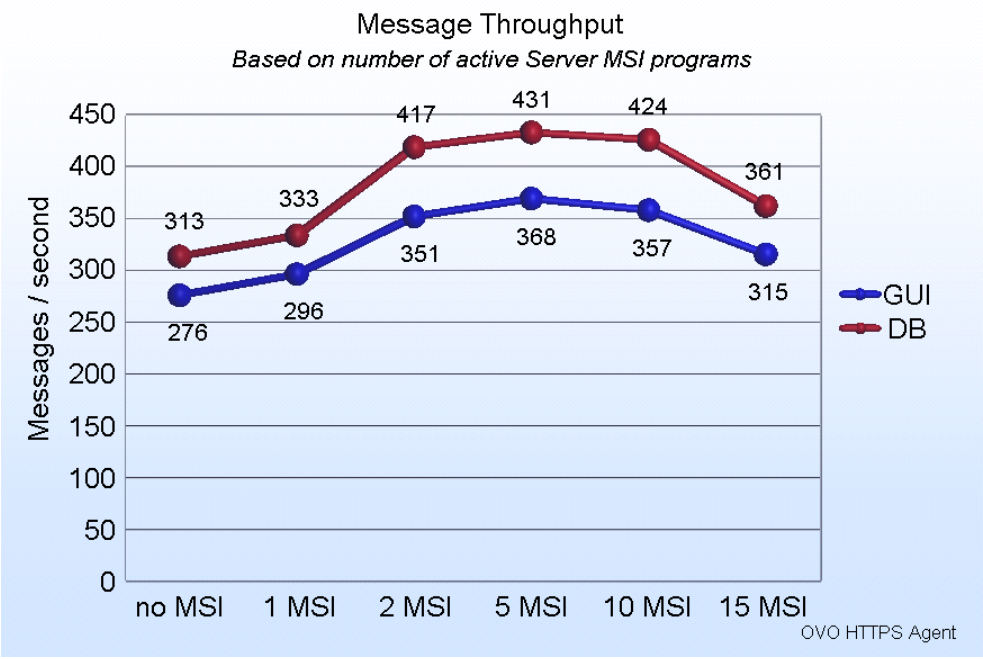


Figure 9 shows the message throughput when diverting messages to server MSI (HTTPS agent) programs on HPOM for UNIX 8.21.

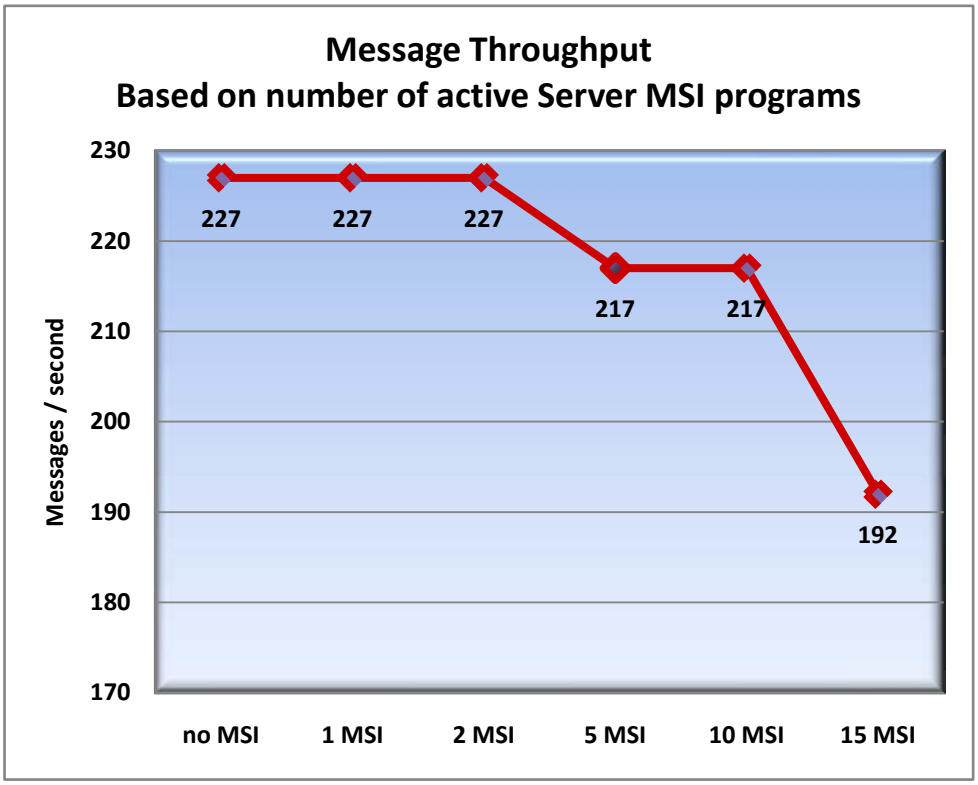
Figure 9 HPOM for UNIX 8.21 Message Throughput – Server MSI



HPOM for UNIX 8.30

Figure 10 shows the results of copying messages to server MSI programs on HPOM for UNIX 8.30.

Figure 10 HPOM for UNIX 8.30 Message Throughput – Server MSI



Interpretation

Based on the test results, the following facts can be derived:

- Using MSI on the agent shows that the parallel registered MSI programs are able to create duplicate OM messages faster than using the OM Message Interceptor (opcmsgi). Also the processing on the HPOM server is faster due to the similarity of the OM messages, because all duplicates are for the same node.
- At HPOM for UNIX 8.21: Using MSI on the server shows that the duplicate messages generated by the parallel MSI programs can be processed faster than messages submitted by the OM HTTPS agent. Therefore the average OM message processing performance is higher compared to scenario where no MSI was used.
- At HPOM for UNIX 8.30: Using the MSI on the server shows that there is almost no performance impact in case only a few MSI programs are used in “copy” mode. In case 5 or more concurrent MSI programs are in use, a slight performance loss is seen.
- Maximum message throughput is nearly identical for agent-based MSI and server-based MSI programs.

Conclusions

Based on the test results, the following conclusions can be drawn:

- Using a few MSI programs results in no performance loss.
- Number of simultaneous server-based MSI programs should not be too high.

Test: Detailed Inspection of Message Processing

This test inspected messages processing in detail on HPOM for UNIX 8.21 only.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Number of messages in the message manager queue `msgmgrq` on the HPOM server over time if a large number of messages were generated on the managed nodes
- **Parameters**
  - Number of HPOM agents
  - HP Operations configuration variables

Scenario

This test was conducted for the following scenario:

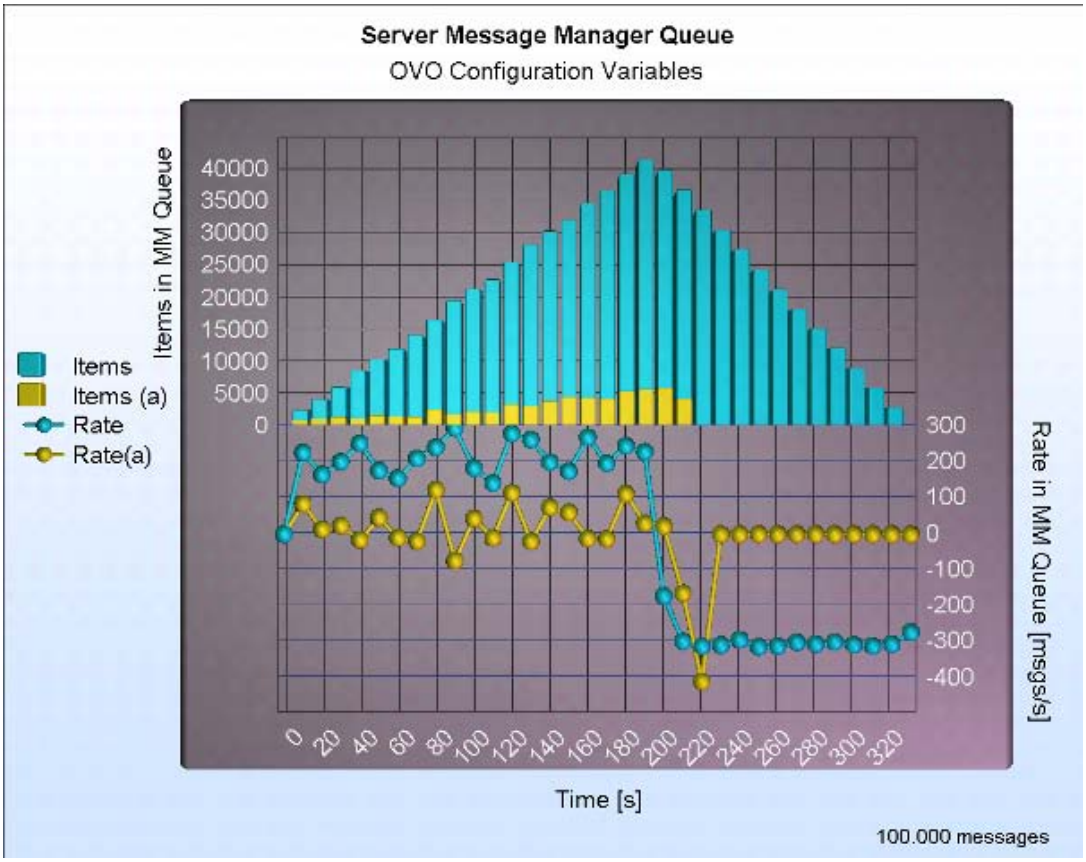
- **Measured values**
  - Number of messages in the message manager queue over time until the queue was empty.
- **Message Generator**
  - 100,000 messages with a severity of normal were used.
  - Messages were targeted to 100 nodes in 4 node groups (that is, 50 messages per node).
  - Operators were responsible for 400 nodes.
  - Node Bank had 2,000 managed nodes.
  - No history messages were used.
- **Java GUI options**
  - No Java GUIs were started.
- **Miscellaneous**
  - None.

Results

Figure 11 shows the number of elements in the queue of process `opcmsgm` over time (in seconds), beginning from the start of the message generator. The x-axis shows the time in

seconds, the left y-axis shows the number of messages in the queue, and the right y-axis shows the message change rate in messages per second.

Figure 11 Server Message Manager Queue – Configuration Variables



Legend

- **Items**

  Number of messages for this point of time.

- **Rate**

  Message rate in messages per second computed using the current and the previous data point.

- **No Option**

  No HPOM variables were set for this test.

- **Option (a)**

  HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD = TRUE`.

Interpretation

Based on the test results, the following facts can be derived:

- In a generic HPOM for UNIX environment (that is, in an environment with no performance configuration variables set) one HTTPS agent can send messages faster than the HPOM server can process them.

- One HTTPS agent is able to send up to 500 messages per second (using the HPOM agent `opcmsg()` C API). This rate was verified in a separate test in which only the agent sent messages, and the HPOM server only queued messages.
- Separate tests showed that, with two HTTPS agents, it is possible to send more messages per second than the server can process. As a result, the queue of the server message manager grows. This principle holds for proxied nodes. For real nodes, a slightly higher throughput can be expected. It was evident that, for a small number of agents, the performance of the message receiver `opcmsgrb` was nearly linear in the number of agents.
- After all the messages were queued, the message processing rate of the HPOM server was as follows:

Configuration Variable	Message Processing Rate
none	310 msgs/s
OPCMSGM_USE_GUI_THREAD = TRUE	410 msgs/s

Conclusions

Based on the test results, the following conclusions can be drawn:

- Use the configuration variable `OPCMSGM_USE_GUI_THREAD = TRUE`.
- If message bursts are not avoidable, make sure that the time of silence is long enough to process all queued messages.
- In most cases, the agent-to-server transfer is faster than the message processing speed on the server.

Test: Message Storm

This test measured the impact of message storms on HPOM for UNIX 8.30 only.

Synopsis

This test computed the following metrics:

- **Metrics**

  Time until the message manager queue (`msgmgrq`) on the HPOM server was cleared following a message burst

Scenario

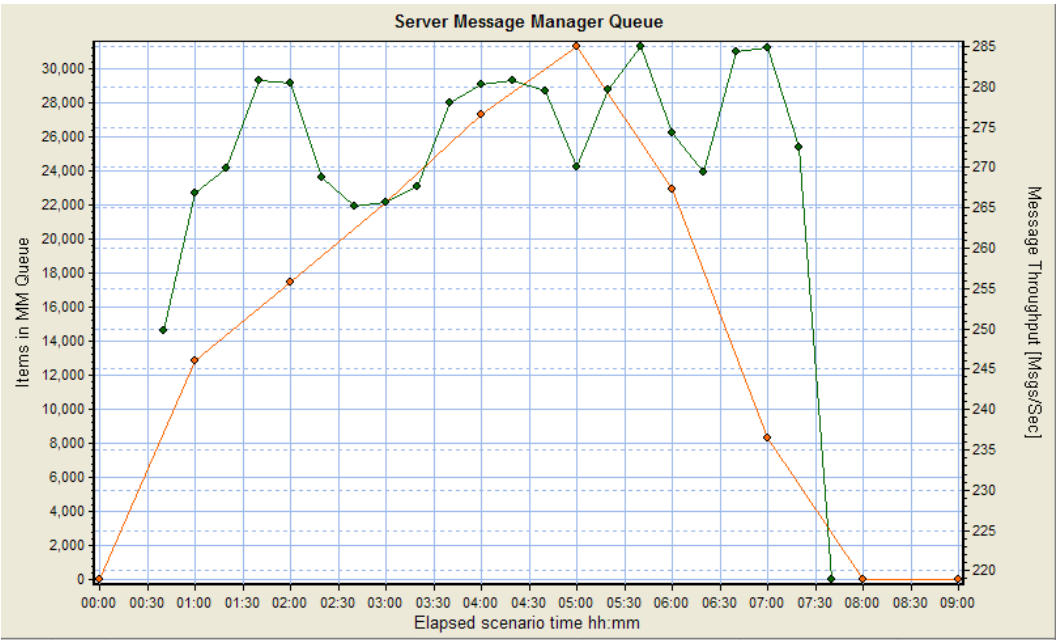
This test was conducted for the following scenario:

- **Measured values**
  - Number of messages in the message manager queue over time until the queue was empty
  - Time until the message manager queue was empty
- **Message Generator**
  - 10,000 and 100,000 normal (`NO`) messages with a random severity were used.
  - Messages were targeted to 100 nodes in 4 node groups (that is, 100 messages per node).
  - Node Bank had 2,000 managed nodes.
  - No history messages were used.
- **Java GUI options**
  - No Java GUIs were started
- **Miscellaneous**
  - HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.

Results

Figure 12 shows the number of elements in the queue of the process `opcmsgm` over time, beginning from the start of the message generator. The x-axis shows the time, the left y-axis shows the number of messages in the queue, and the right y-axis shows the message throughput in messages per second. This figure shows the run when injecting 100,000 messages.

Figure 12 Server Message Manager Queue – Message Throughput



Color	Scale	Measurement	Minimum	Average	Maximum	Std. Deviation
	1	Items In MM Queue	0	14233.7	31307	11213.33
	1	Message Throughput	120.6	272.871	304.2	21.956

Interpretation

Based on the test results, the following facts can be derived:

- Message injection rate from the agents to the server was 360 messages per second.
- Time to recover from all messages was 412 seconds when the message throughput went down to zero. The items in the queue were measured every minute. As a result, a value of zero was displayed after 8 minutes.
- Performance of the message receiver (`opcmsgrb`) was better than the performance of the message manager (`opcmsgm`).
- Peak of the message throughput was reached when the message manager queue had more than 4,000 messages. The various tests showed that, the overall throughput was higher when processing 100,000 messages than when processing 10,000 messages. The average throughput (measured every 5 seconds) when processing 100,000 messages was 272 messages per second with a maximum of 304 messages and a minimum of 120 messages per second. The average throughput when processing 10,000 messages was 218 messages per second.
- Variations in the OM message processing performance is caused by several house keeping tasks of the HPOM for UNIX server processes and Oracle database – for example re-organization of the Message manager queue file and Oracle redo-log switches.

Conclusions

Based on the test results, the following conclusions can be drawn:

- HPOM server is able to survive massive event storms, and then to continue normal operations after a short period of time
- When message bursts are not avoidable, the time of silence should be long enough to process all messages
- Transfer rate from the agents to the server is faster than the message processing rate on the server

Test: MoM Message Forwarding

This test measured Manager-of-Manager (MoM) message forwarding on HPOM for UNIX 8.30 only.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Time until all messages were stored in the Oracle Database on the source and destination HPOM server
  - Time to synchronize between the source and destination servers when acknowledging messages using the Java GUI
- **Parameters**
  - Number of acknowledged messages

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time until all messages were in the database of the source manager
  - Time until all messages were in the database of the destination manager
  - Time until all messages were acknowledged on the destination server Java GUI
- **Message Generator**
  - 10,000 normal (`NO`) messages with a random severity were targeted to the source server.
  - Node Bank had 2,000 managed nodes.
  - No history messages were used.
- **Java GUI options**
  - One Java GUI was started on each server.
  - Different HPOM for UNIX user accounts were used.
  - All messages were shown.
  - Refresh interval was set to five seconds.
- **Miscellaneous**

  HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.

Results

Table 14 lists the message throughput in messages per second with a varying number of CMAs. No Java GUIs were started.

Table 14 Message Throughput – MoM

Forwarding Type	Source		Destination	
	Time [sec]	Msgs/sec	Time [sec]	Msgs/sec
1. With forwarding	83	120	170	59
2. No forwarding	44	227	n/a	n/a

1. Source server kept the notification message. The destination server received the switch control message.

2. Time needed on the source server to process 10,000 messages. No forwarding was in effect.

Table 15 lists the time in seconds to synchronize between the source and destination servers when acknowledging messages.

Table 15 Acknowledging Messages – MoM

Time [Sec]	Number of Acknowledged Messages			
	1000	2000	3000	4000
Synchronization	12	21.37	31.34	38.54
Per 1,000 messages	12	10.685	10.447	9.635

Interpretation

Based on the test results, the following facts can be derived:

- Forwarding messages result in a performance loss.
- Time to synchronize is linear, based on the number of acknowledged messages.

Conclusions

Based on the test results, the following conclusions can be drawn:

- To minimize performance risk, only messages that are needed on the destination server should be forwarded.
- Time to acknowledge a large number of messages can be simply extrapolated after a reference acknowledgment is timed.

Java GUI Start-Up Time

This section describes tests that measured how long it took for the Java GUI to start up fully.

Test: Varying the Number of Managed Nodes and Messages

This test measured Java GUI start-up times when the number of managed nodes and messages were varied on HPOM for UNIX 8.21 and 8.30.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - *HPOM for UNIX 8.21 only*
    - Start-up time of 2 Java GUIs
    - Start-up time of 2 Motif GUIs
  - *HPOM for UNIX 8.30 only:*
    - Start-up time of 1, 5, 10, and 20 Java GUIs
- **Parameters**
  - *HPOM for UNIX 8.21 and 8.30*
    - Number of managed nodes for which users were responsible
    - Number of active messages
  - *HPOM for UNIX 8.30 only*
    - Number of Java GUIs

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time until the last GUI of a type was started and fully functional.
- **Message Generator**
  - 5,000 and 50,000 messages with a severity of normal were generated before the test started.
  - Messages were targeted to 100 nodes in 4 node groups (that is, 50 and 500 messages per node).
  - Operators were responsible for 500 and 2,000 nodes.
  - Node Bank had 10,000 managed nodes.
  - No history messages were included.
-

Java GUI options

- Different HPOM for UNIX user accounts were used.
- All messages were shown.
- Refresh interval was set to 5 seconds.
- HTTPS communication was used.

- **Miscellaneous**
  - HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.

Results

This section describes the results of varying the number of managed nodes and messages on HPOM for UNIX 8.21 and 8.30.

The measured log-on time is the time until all active messages are loaded into the memory of the server side GUI processes (`opcuiwww`).

Difference between the log-on time and the start-up time is mainly the transfer delay of the message buffer from the opcuiwww processes to the Java GUIs.

HPOM for UNIX 8.21

Table 16 lists the results of the test on HPOM for UNIX 8.21. It includes the time in seconds needed to start two Java and two Motif GUIs with different HPOM for UNIX user accounts.

Table 16 GUI Startup Time – Managed Nodes / Active Messages

	Responsible Nodes			
	500		**2,000**	
Active Messages	**Motif**	**Java**	**Motif**	**Java**
5,000	45	6	154	13
50,000	57	68 / 75	167	49 / 89

The tests of the different GUI types were conducted separately:

- **Motif GUIs**

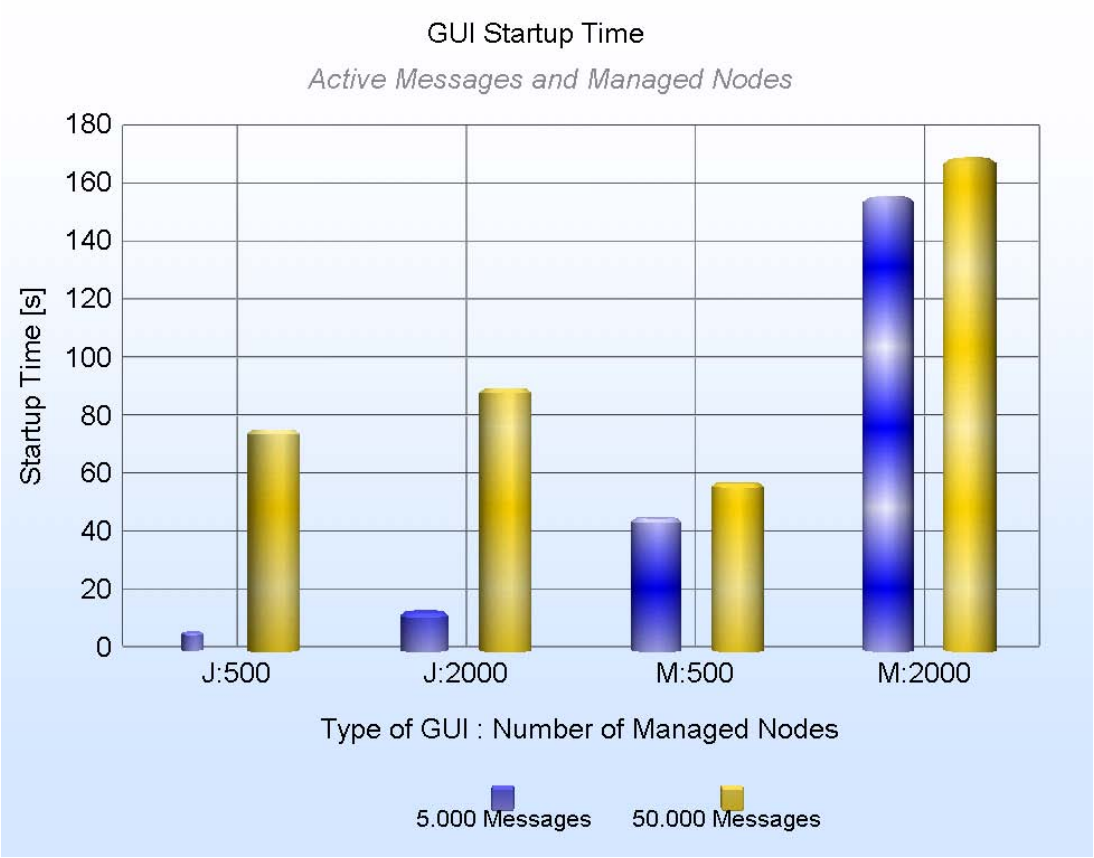
  The start-up time for the second GUI startup was measured. For this reason, the time required to build the map cache was not included.

- **Java GUIs**

  There was a significant difference between the startup time for the first and the second GUI, where 50,000 active messages were shown. The startup time for the first GUI is listed first. In addition, the times for the Java GUIs varied over a range.

Figure 13 shows the bar charts for the tests, which varied the number of managed nodes for which HPOM for UNIX users were responsible, and varied the number of active messages.

Figure 13 Java GUI Startup Time – Managed Nodes and Active Messages



HPOM for UNIX 8.30

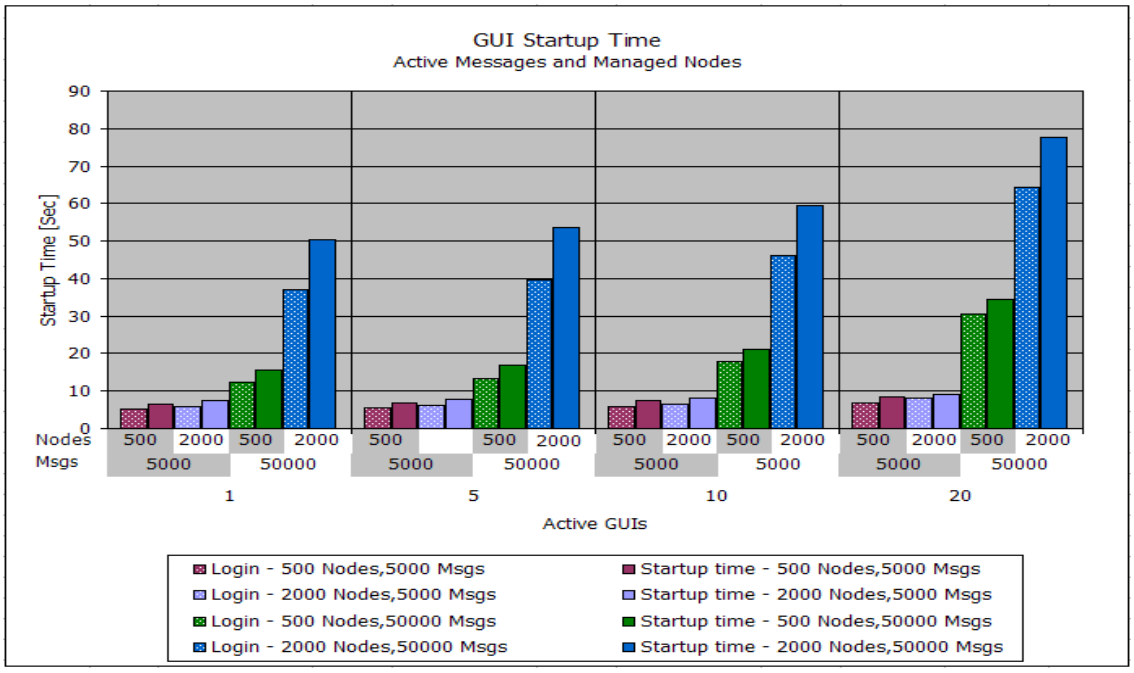
Table 17 lists the time, in seconds, needed to start a varying number of Java GUIs on HPOM for UNIX 8.30. Each time, the GUIs were started with a varying number of nodes and active messages. Each test was conducted separately.

Table 17 GUI Startup Time – Active Messages and Managed Nodes

Active Messages	**5,000**				**50,000**			
Managed Nodes	**500**		**2,000**		**500**		**2,000**	
Active GUIs	**Logon**	**Start-Up Time**	**Logon**	**Start-Up Time**	**Logon**	**Start-Up Time**	**Logon**	**Start-Up Time**
1	5.1	6.609	5.821	7.413	12.201	15.662	36.975	50.498
5	5.438	6.906	6.193	7.771	13.481	17.028	39.638	53.531
10	5.856	7.39	6.42	8.004	17.869	21.272	46.025	59.597
20	6.735	8.61	8.021	9.201	30.653	34.283	64.237	77.618

Figure 14 displays the results of these tests graphically.

Figure 14 GUI Startup Time – Active Messages and Managed Nodes



Interpretation

Based on the test results, the following facts can be derived:

- Number of managed nodes in the HPOM for UNIX user's realm has a significant impact on the start-up time of Motif GUIs.
- Number of active messages has an impact on the start-up time of Java GUIs. However, a significant increase is seen only in the range of 50,000 messages and more.
- Number of managed nodes has no significant impact on the start-up time of the Java GUIs. A significant impact is observed only in the range of 50,000 messages, where 2,000 nodes make the difference between an acceptable data buffer size and a bottleneck
- Number of Java GUIs has a very small impact on the start-up time of the Java GUIs for HPOM for UNIX 8.30.
- Most of the time is consumed by the logon (as seen in previous tests), where all active messages are loaded into the memory of the `opcuiwww` process.

Conclusions

Based on the test results, the following conclusions can be drawn:

- Number of active messages should be kept as low as possible to enable a faster start-up time (for example, by using duplicate suppression, event correlation, and so on).
- If the Motif GUI is used, the number of managed nodes in the HPOM for UNIX user's realm should be kept as small as possible.
- Configuration "Show All Messages" should be used only if absolutely needed in case you have a huge amount of active messages.
- Only the number of Java GUI's needed at the moment should be started.

Test: Varying the Number of CMAs

This test measured Java GUI start-up times when the number of CMAs was varied on HPOM for UNIX 8.21 and 8.30.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - *HPOM for UNIX 8.21 only*

    Start-up time of two Java GUIs displaying all CMAs attached to the HPOM messages
  - *HPOM for UNIX 8.30 only*

    Start-up time of varying number of Java GUIs until all messages with CMAs were displayed
  - *HPOM for UNIX 8.30 only*

    Number of messages per second (message throughput) until all messages was stored in the Oracle Database

- **Parameters**
  - Number of CMAs attached to the messages
  - Number of active Java GUIs

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Number of messages stored in the Oracle Database divided by the receive time elapsed was measured as message throughput.
  - Time until the last Java GUI was up and functional (that is, until messages were displayed in the message browser). The Java GUI was configured in such a way that all CMAs were shown in the browser.
- **Message Generator**
  - 10,000 messages with a severity of normal were generated before the test started.
  - Varying number of CMAs was attached to each message. Each CMA had a length of 20 characters.
  - Messages were targeted to 100 nodes in 4 node groups (that is, 100 messages per node).
  - Operators used were responsible for 2,000 nodes.
  - Node Bank contained 10,000 managed nodes.
  - No history messages were used.
- **Java GUI options**
  - Different HPOM for UNIX user accounts were used.

- All messages were shown.
- Refresh interval of 5 seconds was used.

- **Miscellaneous**
  - HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.

Results

This section describes the results of varying the number of CMAs on HPOM for UNIX 8.21 and 8.30.

HPOM for UNIX 8.21

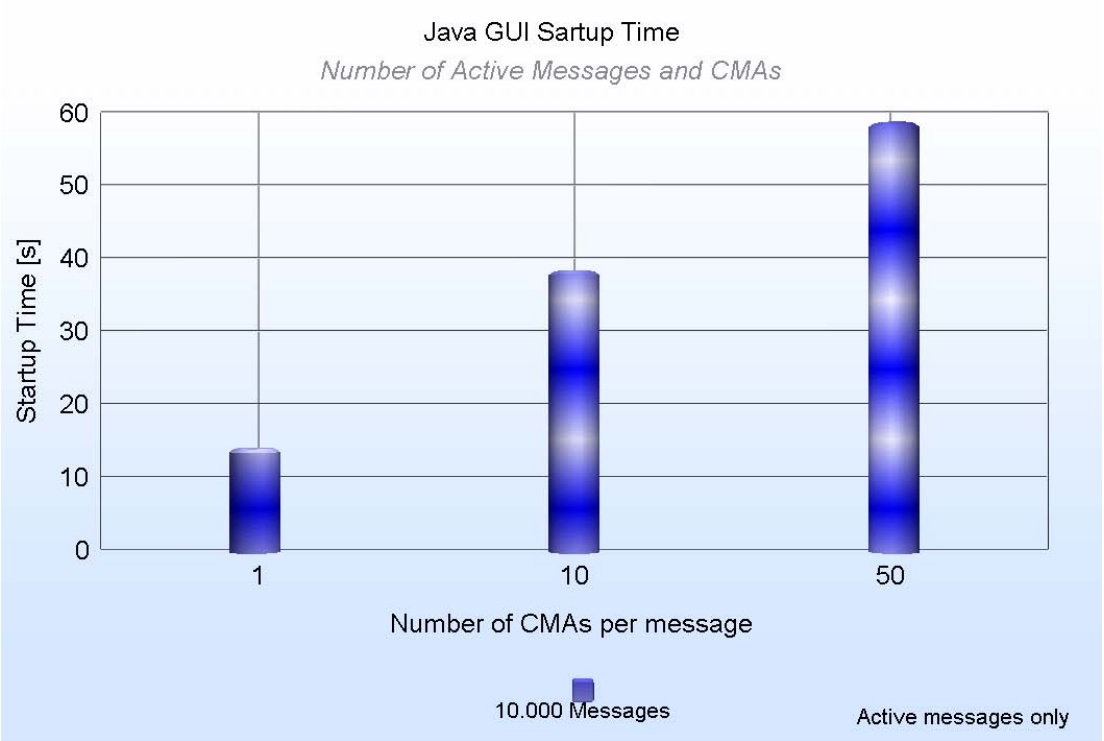
Table 18 lists the results of this test on HPOM for UNIX 8.21. It includes the time in seconds needed to start two Java GUIs with different HPOM for UNIX user accounts, with a varying number of active messages (and CMAs), and with no history messages.

Table 18 Java GUI Start-Up Time – CMAs in Active Messages

Active Messages	Number of CMAs		
	1	**10**	**50**
10,000	14.5	38	58

Figure 15 displays the results of this test graphically. No graph is shown for the test with history messages only.

Figure 15 Java GUI Start-Up Time – CMAs



HPOM for UNIX 8.30

Table 19 lists the message throughput, in messages per second, for a varying number of CMAs on HPOM for UNIX 8.30. No Java GUIs were started.

Table 19 Message Throughput – Messages with CMAs

Message Throughput	Number of CMAs		
	1	10	50
Messages per second	212.8	192.3	112.4

Table 20 lists the time, in seconds, needed to start a varying number of Java GUIs, with a varying number CMAs.

Table 20 Java GUI Startup time – Messages with CMAs

Active GUIs	Number of CMAs					
	1		10		50	
	Logon	Start-Up Time	Logon	Start-Up Time	Logon	Start-Up Time
1	9.987	13.481	12.378	20.266	23.292	51.038
5	10.372	13.882	12.75	21.134	24.275	54.258
10	10.606	14.212	13.101	22.09	24.548	57.659
20	14.73	19.618	20.96	29.058	36.357	73.137

Figures 16a and 16b display the results of these tests graphically.

Figure 16a Java GUI Logon/Startup Time – CMAs

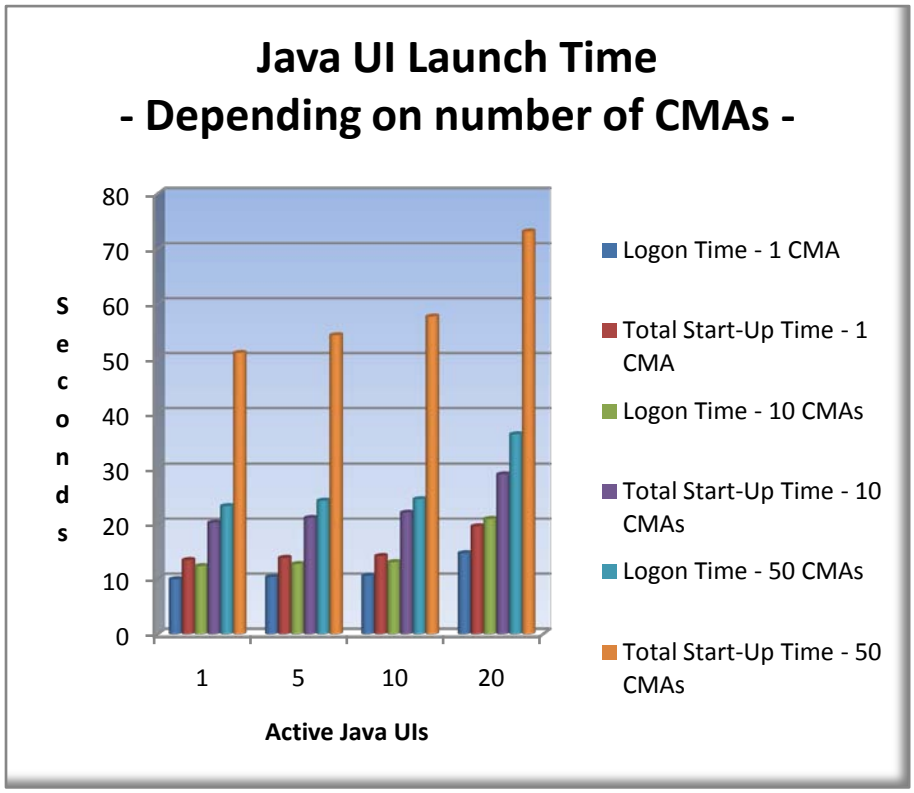
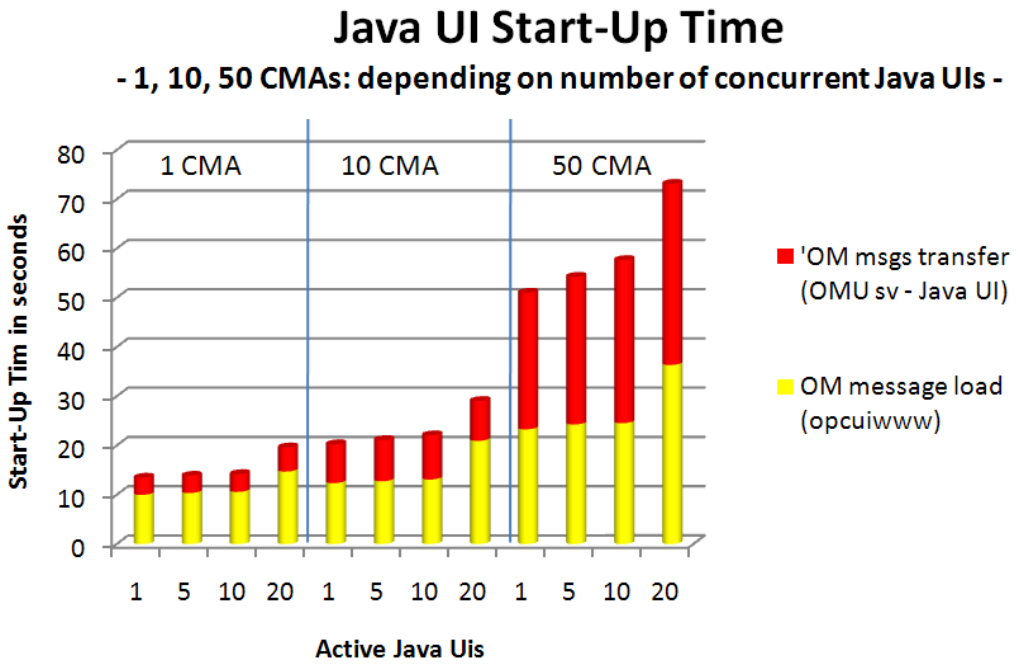


Figure 16b Java GUI Startup Time Breakdown – CMAs



Interpretation

Based on the test results, the following facts can be derived:

- Number of custom message attributes per message has an noticeable impact on the start-up time of the Java GUI
- Number of Java GUIs has a small impact on the start-up time. The impact is non-linear.
- Delay in the start-up time of the Java GUI depends on the number of CMAs per message, in a non-linear fashion.
- Log-on time consumes most of the total start-up time, except in the 50 CMAs. The log-on time is the time until all active messages are loaded into the memory of the server side GUI processes (`opcuiwww`).
- Difference between the log-on time and the start-up time is mainly the transfer delay of the message buffer from the opcuiwww processes to the GUIs. The anomaly in the 50 CMAs results from the large size of the buffer, in which it takes more time to transfer to the GUI over the network.
- The HPOM for UNIX 8.30 Java UI starts up significantly faster than 8.21.

Conclusions

Based on the test results, the following conclusions can be drawn:

- Performance cost of the start-up time of the Java GUI and the message throughput when using CMAs is noticeable and non-linear.
- Number of CMAs should be kept at or below 10 CMAs per message.

Test: Message Acknowledgment

This test measured the time needed to acknowledge messages using the Java UI with HTTPS communication protocol.

The test ran on HPOM for UNIX 8.30 only.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Time needed to acknowledge multiple messages using the Java GUI
- **Parameters**
  - Number of Java GUIs
  - Number of acknowledged messages
  - Internal handling of the acknowledged message (move or mark)

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time it took for the last message to disappear on all started Java GUIs when messages were acknowledged on one Java GUI.

- **Message Generator**
  - 50,000 normal (`NO`) messages a random severity and random number of CMAs (1, 10 or 50) were generated prior to each test.
  - Each CMA had a name length of 20 characters and a value length of 50 characters.
  - No messages were generated during the test.
  - Messages were targeted to 100 nodes in 4 node groups.
  - Node Bank contained 10,000 managed nodes.
  - No history messages were used.
- **Java GUI options**
  - Different HPOM for UNIX user accounts were used.
  - All messages were shown.
  - Refresh interval was set to 5 seconds.
- **Miscellaneous**
  - HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.
  - HPOM message mover process was disabled (that is, the HPOM for UNIX parameter `OPC_ACK_MOVE_INTERVAL` was set to `0`).
  - `OPC_DIRECT_ACKN_LIMIT` was set to `0` for mark only.

Results

Tables 21 and 22 list the time, in seconds, it took for the last message to disappear from the last Java GUI message browser when a varying number of messages were acknowledged.

Table 21 lists the rate at which messages were acknowledged and moved to a history table.

Table 21 Message Acknowledgement – Move

	Number of Acknowledged Messages		
Active GUIs	**1000**	**5000**	**10000**
1	24.11	38.89	64.25
5	29.07[5]	62.76	98.34

Table 22 lists the rate at which messages were acknowledged and marked but not moved.

Table 22 Message Acknowledgement – Mark Only

	Number of Acknowledged Messages		
Active GUIs	**1000**	**5000**	**10000**
1	9.85	33.2	33.9
5	31.98	59.62	95.5

[5] The anomaly that the acknowledgement of 1,000 messages with 1 active Java UI was faster in “move” mode was caused by regular house keeping tasks, which just happened in this test run, such as DB redo log switch, Java GUI refresh interval.

Interpretation

Based on the test results, the following facts can be derived:

- At the 10,000 messages range, a high network activity was observed.
- Marking messages on acknowledgement was faster than moving them.
- When moving 10,000 messages and using 5 GUIs, the disk utilization reached 97%.
- The absolut performance gain for using “mark only” vs. “move” get less the more Java GUIs run in parallel, because the faster acknowledgement on the DB level is not so big, because most processing time is spent in the Java GUIs and the Java GUI background processes (opcuiwww).

Conclusions

Based on the test results, the following conclusions can be drawn:

- For optimizing the usability of the Java GUI, acknowledged messages should be marked only, and then later moved to the history table automatically.
- To avoid disk and network bottlenecks, no more than 5,000 messages should be acknowledged at a time.

Service Navigator Start-Up Time

This section describes tests of Service Navigator start-up times when the shape and size of the service tree was varied.

The test used trees with the following attributes:

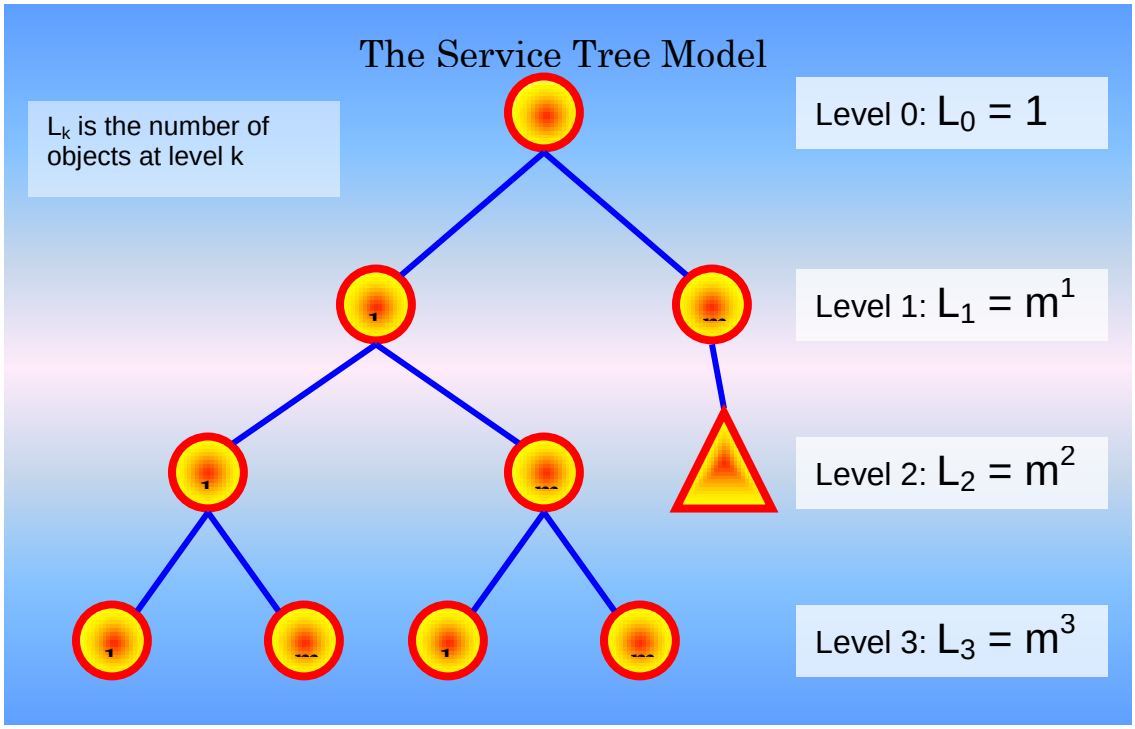
- Varying depth and width
- No use factor or links (that is, associations from objects to their nephews, or children of siblings)

The number of objects at level k is $L_k = m^k$ where $k = 0$ for the root level.

The total number of objects in the tree $\frac{m^{n+1}-1}{m-1}$ where:

- m is the breadth (number of children for each object).
- n is the depth (number of levels below the top level), as shown in Figure 17.

Figure 17 Service Tree Model



Test: Varying the Shape and Size of the Service Tree

This test measured Service Navigator start-up time when the shape and size of the service tree were varied without the "Service Load on Demand" (SLOD) feature.

Tests ran on HPOM for UNIX 8.21 only.

For test results with the "Service Load on Demand" feature, please refer to the "OM8.10 – Performance Guide". These tests have been not repeated, since the shape and size of the service tree has very minimal impact when using the SLOD feature.

In most cases, using SLOD is the preferred option!

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Time needed to start two Java GUIs for different HPOM for UNIX users assigned to the same service tree.
- **Parameters**
  - Number of levels in the service tree (depth)
  - Number of children for each intermediate object in the tree (breadth)
  - Feature "Service Load on Demand" switched off

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Start-up time of the Java GUI in situations where a service tree was assigned to the HPOM for UNIX user starting the GUI (and therefore the start-up time of Service Navigator).
- **Message Generator**
  - No messages were generated or used.
- **Java GUI options**
  - Different HPOM for UNIX user accounts, but the same service tree, were used.

Results

Table 23 lists the total number of objects in the service tree, depending on the number of levels and the breadth of the tree. The configurations shown in **boldface** (B/D = 10/10, B/D = 40/5, B/D = 40/10) were not tested. The configurations shown in *italics* (B/D = 3/10, B/D = 40/3) were used to compare the impact of the shape of the tree for two trees with a nearly identical total number of services.

Table 23 Service Navigator – Number of Objects in Tree

	Depth of Tree		
Breadth of Tree	**3**	**5**	**10**
3	40	364	*88573*
10	1111	111111	**1111E+10**
40	*65641*	**105025641**	**1,0755E+16**

Table 24 shows the times needed to start the *first* Java GUI with Service Load on Demand (SLOD) switched off.

Table 24 First Java GUI: Service Navigator – Start-Up Times with No SLOD

	Depth of tree		
Breadth of tree	**3**	**5**	**10**
3	5	6	272
10	8	222	
40	40		

Table 25 shows the times needed to start the *second* Java GUI with Service Load on Demand (SLOD) switched off.

Table 25 Second Java GUI: Service Navigator – Start-Up Times with No SLOD

	Depth of tree		
Breadth of tree	**3**	**5**	**10**
3	5	7	331
10	11	325	
40	67		

Interpretation

Figure 18 shows the result of this test with one Java GUI running. The bars marked with the yellow arrow show the results of configurations with a nearly identical total number of objects, but with a different number of levels and with a different breadth.

Figure 18 Service Navigator Start-Up Times – 1 Java GUI

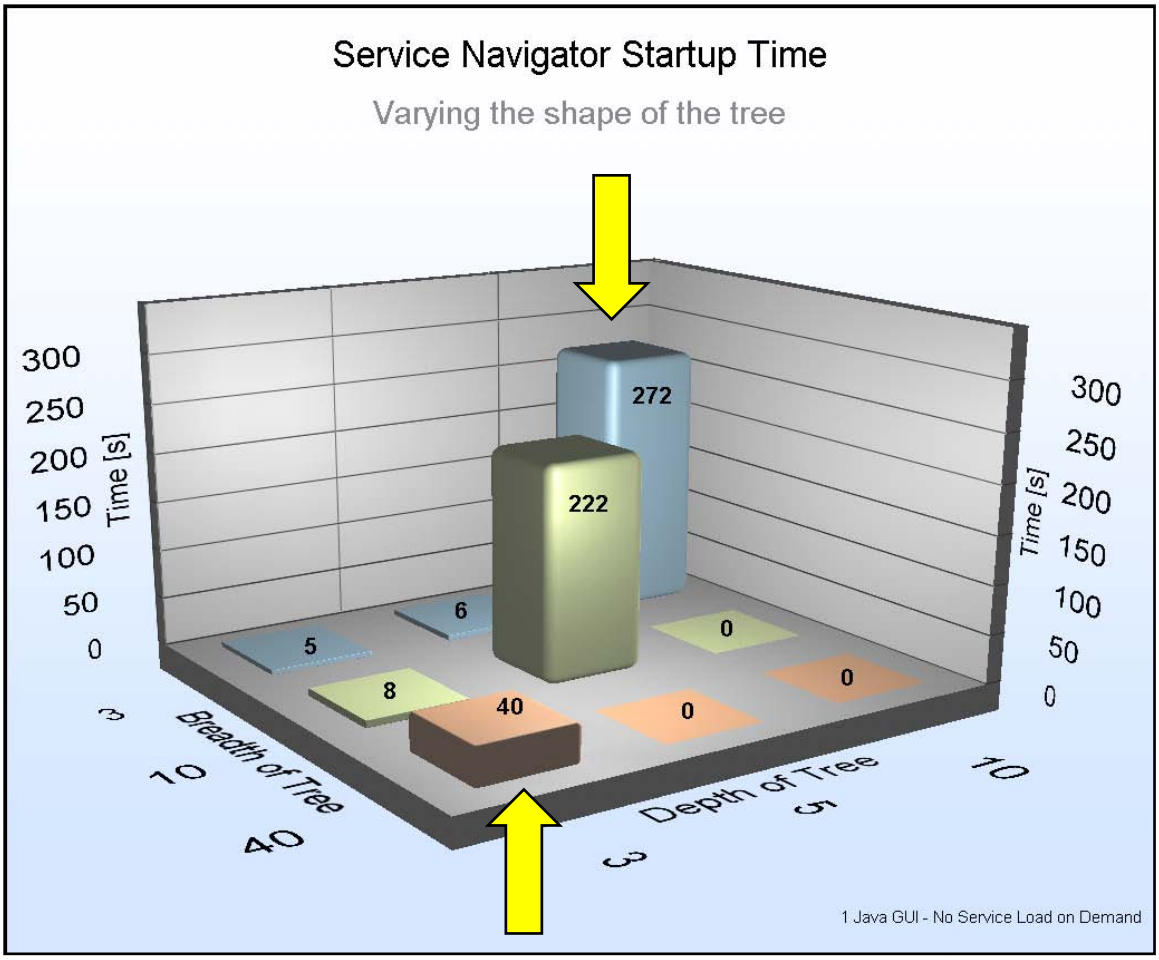
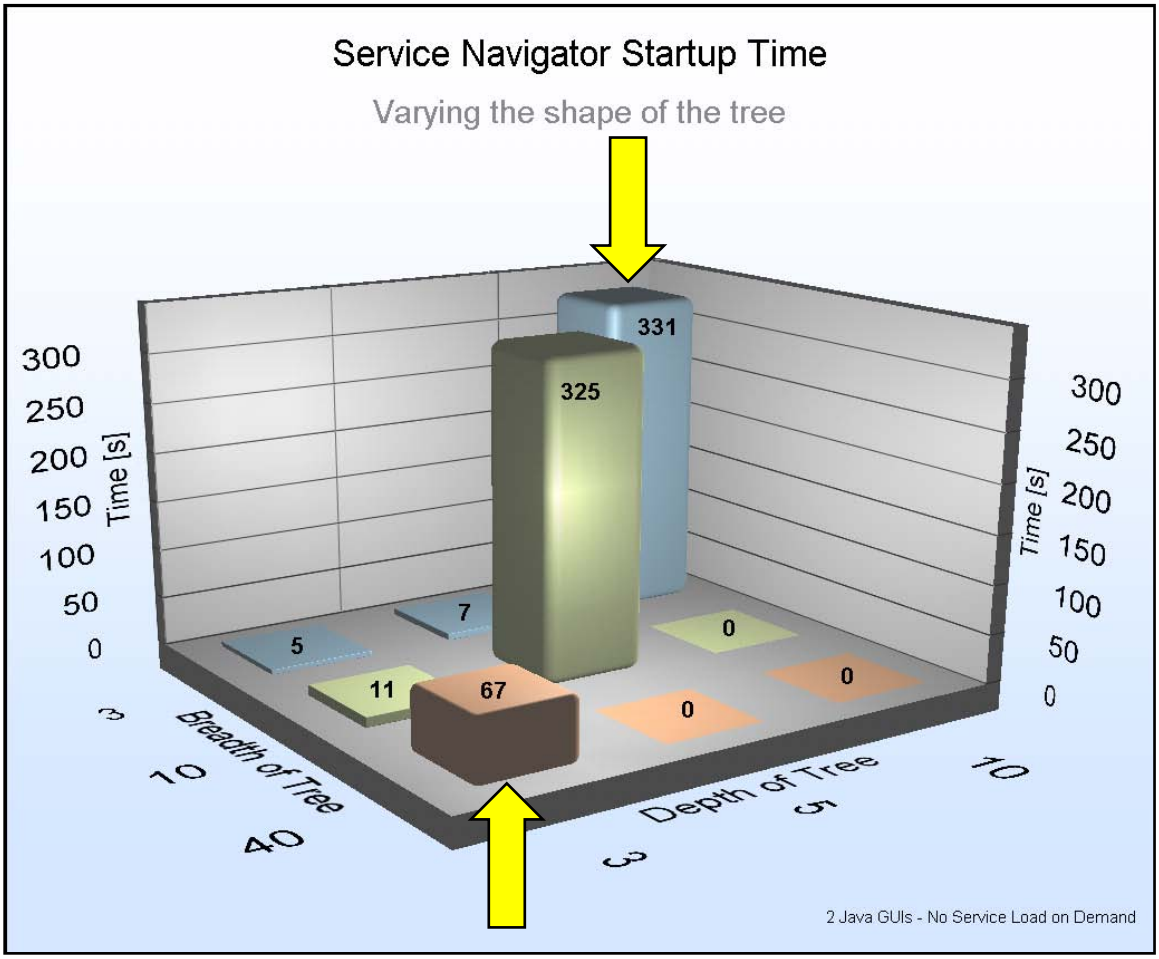


Figure 19 shows the results of this test with two Java GUIs running. The bars marked with the yellow arrows show the results of the configurations with a nearly identical total number of objects, but with a different number of levels and with a different breadth.

Figure 19 Service Navigator Start-Up Times – 2 Java GUIs



Based on the test results, the following facts can be derived:

- Total number of objects is not the only attribute that has an impact on the start-up time. The number of the service levels (the depth of the tree) has the strongest impact on the start-up time. For a comparison of Breadth 3 – Depth 10 with Breadth 40 – Depth 3, see Figures 18 and 19.

Conclusions

Based on the test results, the following conclusions can be drawn:

- If deep service trees (that is, service trees with more than 5 levels) are used, the feature "Service Load on Demand" should be used to avoid long delays in the start-up time of the Java GUI.

History Download Performance

This section describes tests of history download performance.

Test: Downloading 100,000 History Messages

This test measured history download performance of 100,000 history messages for HPOM for UNIX 8.21 and 8.30.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - *HPOM for UNIX 8.21 and 8.30*

    Time to download 100,000 history messages
- **Parameters**
  - Oracle log buffer and log file groups
  - Commit count

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time needed to download a 100,000 history messages.
    The download tool `opchistdwn` was started on the command line.
- **Message Generator**
  - 100,000 history messages with a severity of normal were generated before the test.
  - No active messages existed.
  - No messages were generated during the test.
  - Node Bank contained 10,000 managed nodes.
- **Java GUI options**
  - No Java GUIs were started.
- **Environment**
  - Automatic downloading of history messages was disabled.
  - Default HPOM for UNIX configuration for marking and moving acknowledged messages was used.
  - HPOM message mover process was disabled (that is, `OPC_ACK_MOVE_INTERVAL` was set to `0`).
  - HPOM server configuration variable `OPCMSGM_USE_GUI_THREAD` was set to `TRUE`.
  - HPOM server configuration variable `OPC_UPDWN_COMMIT_COUNT` was set to `1000` for the second part of the test. The default value is `100`.

Results

This section describes the history message download performance for HPOM for UNIX 8.21 and 8.30.

HPOM for UNIX 8.21

Table 26 lists the times, in seconds, to download 100,000 history messages beginning with the start of the `opchistdwn` utility and ending with its termination, on HPOM for UNIX 8.21. The table also shows the number of messages downloaded per second.

Table 26 Performance of History Download – HPOM for UNIX 8.21

Configuration	A	B
Time for download [s]	2,664	247
Messages per second	37.5	405

Legend

Configuration A:

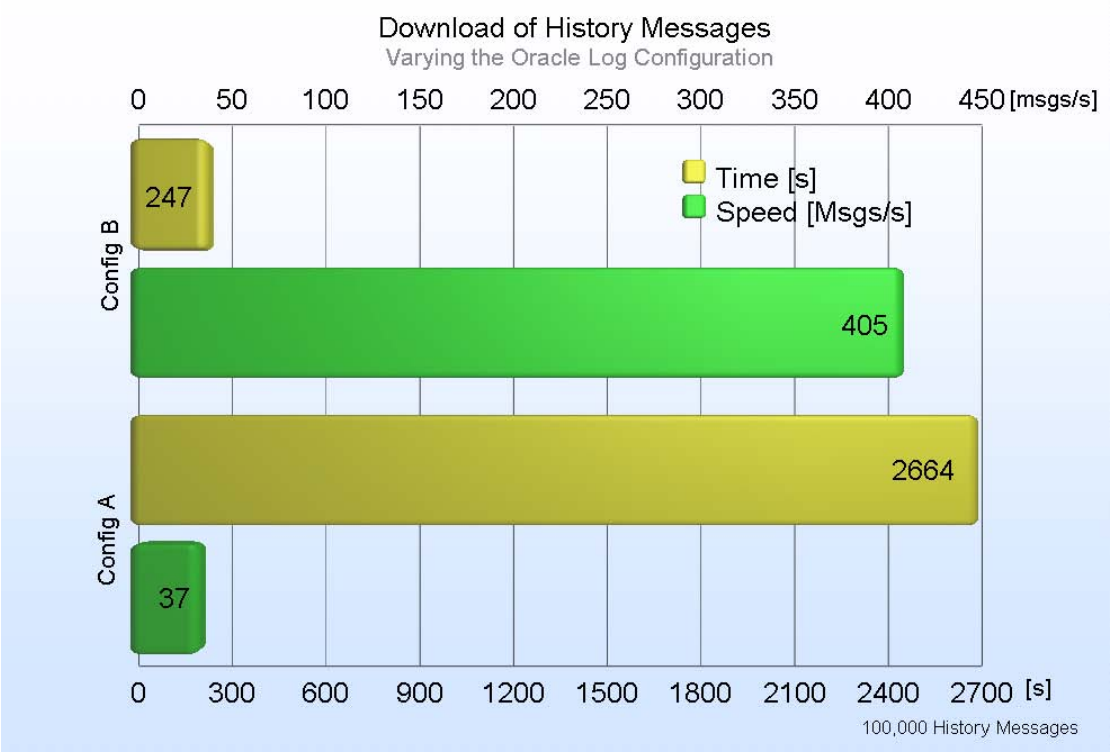
- Oracle log buffer size = 1.5MB
- 3 log file groups, 1 file per group, 20 MB per file

Configuration B:

- Oracle log buffer size = 1.5 MB
- 6 log file groups, 1 file per group, 100 MB per file
- HPOM server configuration variable `OPC_UPDWN_COMMIT_COUNT = 1000`

Figure 20 shows the results of the test on HPOM for UNIX 8.21. It includes the times required to download up to 1,000 messages. The `opchistdwn` process took 50% of one CPU. The Oracle Database disk array was used up to 20%.

Figure 20 Performance of History Download – HPOM for UNIX 8.21



HPOM for UNIX 8.30

Table 27 lists the time, in seconds, it took to download 100,000 messages with a high bulk commit and with the default bulk commit on HPOM for UNIX 8.30. Both tests were performed with the same configuration as the other HPOM 8.30 tests (5 log groups with 100 MB per file).

Table 27 Performance of History Download – HPOM for UNIX 8.30

Commit Count	Not Set (Default 100)	1000
Seconds for download	272.21	192.95
Seconds per 1,000 messages	2.72	1.93
Messages per second	367.36	518.27

Interpretation

Based on the test results, the following facts can be derived:

- Download speed is about 400 to 500 messages per second with HPOM for UNIX 8.30, which is much higher than what has been measured with HPOM for UNIX 8.21.
- Time needed to download of history messages depends on the configuration of Oracle Database parameters, especially the number and size of redo log files.
- As expected, using a higher bulk commit increases the download rate (a 140% gain).
- HPOM server is able to download a high volume of messages in a relatively short period of time.

Conclusions

Based on the test results, the following conclusions can be drawn:

- After a reference download has been timed, the download of history messages can be extrapolated (this conclusion was already found with previous performance tests and therefore we did not repeat downloading different amounts in the 8.21 and 8.30 performance tests).
- When downloading, it is wise to use a higher bulk commit size (`OPC_UPDWN_COMMIT_COUNT`) than the default.
- Increasing the number and size of redo logs will increase history download performance dramatically. 5 redo log files with size 100 MB each seems to be a good combination.

Message Throughput of IP vs. non-IP Nodes

This section describes tests of HPOM for UNIX with real existing and proxied managed nodes.

Test: Message Throughput on IP vs. non-IP Nodes

This test was performed on HPOM for UNIX 8.21 only.
In the usual performance tests, messages for proxied non-IP nodes were created on physical nodes by using the `opcmsg()` API. To simulate the distribution of messages to managed nodes, the messages were targeted to a number of different **non-IP** nodes. These nodes were not defined as external nodes, but individually configured as "non-IP" nodes.

In this test, all of the messages were created on one physical node and targeted to this node. That is, the node parameter of the `opcmsg()` API was not used. All of the messages were first queued on the server. No messages arrived during the timed message processing. The message manager process was working alone on the queues.

Synopsis

This test computed the following metrics and parameters:

- **Metrics**
  - Number of messages per second (message throughput) until all messages were stored in the Oracle Database
- **Parameters**
  - HPOM server configuration variables

Scenario

This test was conducted for the following scenario:

- **Measured values**
  - Time on server to receive all messages (Receive Time of message) divided by number of messages was taken as the rate of messages stored in the Oracle Database.
- **Message Generator**
  - 100,000 messages with a severity of normal were used.
  - Messages were targeted to 1 real node (configured as IP node).
  - All messages were queued on the server. No messages were sent during timed message processing.
  - Operators were responsible for 400 nodes.
  - Node Bank contained 2,000 managed nodes.
  - No history messages were used.
- **HPOM server options**
  - `OPCMSGM_USE_GUI_THREAD = TRUE` [6]

[6] HPOM server options were set with ovconfchg –ovrg server –ns opc –set <variable> <value>.

Results

Table 28 lists the message throughput for the different scenarios.

Table 28 Message Throughput – HPOM for UNIX 8.21

Platform	Configuration (Messages/Second)	
	Suppress Duplicates OFF	**Suppress Duplicates ON No Messages Keys**
IA64 – Real	480	64
IA64 – Proxied	410	28

Legend

IA64 – Real: Platform was IA64. All messages were targeted to one real managed node

IA64 – Proxied: Platform was IA64. All messages were distributed to proxied non-IP nodes.

Interpretation

Based on the test results, the following facts can be derived:

- Message throughput is higher for real nodes (configured as individual IP nodes) – although it has to be considered that, in the test, only one real node was used. In typical environments, there are many real nodes running an HPOM agent generating the messages and acting as the source of these messages.
- Message duplicate suppression works faster for real (IP) managed nodes

In the test, only one real node was used. In typical environments, there are many real nodes that generate the messages and act as the sources of those messages.

Conclusions

Based on the test results, the following conclusions can be drawn:

- Message throughput in real environments, in which more real nodes (configured as IP node in HPOM and running an OM agent) than proxied nodes are used, can be expected to be higher than in this simulated environment.

Appendix A: Tuning the VXFS File System

The parameters for the database file system on HP-UX 11.23 were tuned according to recommendations from performance specialists.

The following file was changed:

```
/etc/vx/tunefstab
```

The changes were applied using the following command:

```
vxtunefs –s /dev/vg01/ovodb
```

The results looked like this:

```
system_default read_pref_io=65536
system_default read_nstream=4
system_default read_unit_io=65536
system_default write_pref_io=65536
system_default write_nstream=4
system_default write_unit_io=65536
system_default pref_strength=10
system_default buf_breakup_size=131072
system_default discovered_direct_iosz=262144
system_default max_direct_iosz=1048576
system_default default_indir_size=8192
system_default qio_cache_enable=0
system_default write_throttle=0
system_default max_diskq=1048576
system_default initial_extent_size=1
system_default max_seqio_extent_size=2048
system_default max_buf_data_size=8192
system_default hsm_write_prealloc=0
system_default read_ahead=1


/dev/vg01/ovodb read_pref_io=65536
/dev/vg01/ovodb read_nstream=64
/dev/vg01/ovodb read_unit_io=65536
/dev/vg01/ovodb write_pref_io=65536
/dev/vg01/ovodb write_nstream=64
/dev/vg01/ovodb write_unit_io=65536
/dev/vg01/ovodb pref_strength=30
/dev/vg01/ovodb buf_breakup_size=65536
/dev/vg01/ovodb discovered_direct_iosz=16M
/dev/vg01/ovodb max_direct_iosz=5242880
```

```
/dev/vg01/ovodb default_indir_size=16384
/dev/vg01/ovodb qio_cache_enable=0
/dev/vg01/ovodb max_diskq=1G
/dev/vg01/ovodb initial_extent_size=32
/dev/vg01/ovodb max_seqio_extent_size=2048
/dev/vg01/ovodb max_buf_data_size=65536
```

Appendix B: Memory and Kernel Parameter Formulas

The `ovoinstall` script calculates and checks important hardware resources, patches, and kernel parameters during the installation of HPOM for UNIX.

The following tables list the formulas for the memory and kernel parameter requirements that `ovoinstall` uses for your information (as of the 8.31 `ovoinstall` package from October 2008).

Table 29 Kernel Parameters for HP-UX 11.11

Parameter	Formula
fs_async	0
max_thread_proc	1024 + 6 * <#Java Operators>
maxdsiz	0x40000000
maxdsiz_64bit	0x80000000
maxfiles	256 + 3 * (MAX((<#DCE Agents> - 35), 0)
maxssiz	0x8000000
maxssiz_64bit	0x40000000
maxuprc	1024 + 6 * (<#Motif Operators> + <#Java Operators>)
maxusers	256 + 2 * (<#Motif Operators> + <#Java Operators>)
msgseg	32767
ncallout	7500
nfile	4000 + 100 * (<#Motif Operators> + <#Java Operators>)
nflocks	4046 + 5* (<#Motif Operators> + <#Java Operators>)
nkthread	MAX(ncallout, 7500)
nproc	200 + 15 * maxusers + 6 * (<#Motif Operators> + <#Java Operators>)
semmns	8192
semmni	4096
semmnu	semmni - 4
semmsl	400
semume	250
semvmx	32767
shmmax	<RAM size>
shmmni	512
shmseg	400
vps_ceiling	64

Table 30 Kernel Parameters for HP-UX 11.23 and 11.31 (PA-RISC and Itanium)

Parameter	Formula
executable_stack	0
fs_async	0
ksi_alloc_max	nproc * 8
max_thread_proc	1024 + 6 * <#Java Operators>
maxdsiz	0x40000000
maxdsiz_64bit	0x80000000
maxfiles	15 + 3 * <#DCE Agents>
maxssiz	0x8000000
maxssiz_64bit	0x40000000
maxuprc	(9 * nproc) / 10
msgmap	2 + msgmni
msgmni	4096
msgseg	32767
msgtql	4096
ncsize (11.23)	ninode + 1024
ncsize (11.31)	ninode
nfile (PA-RISC)	4000 + 100 * (<#Motif Operators> + <#Java Operators>)
nfile (IA64)	15 * nproc + 2048
nflocks	4046 + 5* (<#Motif Operators> + <#Java Operators>)
ninode	8 * nproc + 2048
nkthread	(nproc * 7) / 4 + 16
nproc (PA-RISC)	4096
nproc (IA64)	200 + 15 * maxusers + 6 * (<#Motif Operators> + <#Java Operators>)
semmni	4096
semmns	semmni * 2
semmnu	semmni - 4
semmsl	400
semume	250
semvmx	32767
shmmax	<RAM size>
shmmni	512
shmseg	400
vps_ceiling	64

Table 31 Memory and Swap Requirements

Parameter	Formula
RAM (MB)	512 + 35 * <#Motif Operators> + 128 * <#Java Operators>
Swap (MB)	<RAM> + 512 + 12 * <#Motif Operators> + 64 * <#Java Operators>