HP Operations Orchestration

Software Version: 9.00

Performance Benchmarking Report

Document Release Date: November 2010

Software Release Date: June 2010

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

For information on open-source and third-party software acknowledgements, see in the documentation set for this release, Open-Source and Third-Party Software Acknowledgements (3rdPartyOpenNotices.pdf).

On the Web: Finding OO support and documentation

There are two web sites where you can find support, patches, and documentation, including updates to OO Help systems, guides, and tutorials. Both sites provide all documentation, including tutorials. In addition:

- OO Support site provides platform patches.
- HP Live Network site (in the OO area) provides content patches.

Support

You can obtain platform patches (that is, patches that fix issues with OO Central, Studio, or other components of OO) and the HP OO documentation set and tutorials from the HP Software Product Manuals web site. You will need an HP Passport to log in to the web site.

To obtain HP OO documentation and tutorials

1. Go to the HP Software Product Manuals web site (http://support.openview.hp.com/selfsolve/manuals).
2. Log in with your HP Passport user name and password.

   OR

   If you do not have an HP Passport, click **New users – please register** to create an HP Passport, then return to this page and log in.

   If you need help getting an HP Passport, see your HP OO contact.
3. In the **Product** list box, scroll down to and select **Operations Orchestration**.
4. In the **Product Version** list, click the version of the manuals that you're interested in.
5. In the **Operating System** list, click the relevant operating system.
6. Click the **Search** button.
7. In the **Results** list, click the link for the file that you want.

BSA Essentials Network

For content patches (that is, patches that fix issues with operations or flows), troubleshooting aids, support contract management, product manuals and more, visit the following site: https://www.www2.hp.com.

This is the **HP Live Network** Web page. To sign in:

1. Click **Login Now**.
2. On the **HP Passport sign-in** page, enter your HP Passport user ID and password and then click **Sign-in**.
3. If you do not already have an HP Passport account, do the following:
   a. On the **HP Passport sign-in** page, click **New user registration**.
   b. On the **HP Passport new user registration** page, enter the required information and then click **Continue**.
   c. On the confirmation page that opens, check your information and then click **Register**.

d. On the **Terms of Service** page, read the Terms of use and legal restrictions, select the **Agree** button, and then click **Submit**.

4. On the **BSA Essentials Network** page, click **Operations Orchestration Community.**

   The Operations Orchestration Community page contains links to announcements, discussions, downloads, documentation, help, and support.

Note: Contact your OO contact if you have any difficulties with this process.

In OO: How to find Help, PDFs, and tutorials

The HP Operations Orchestration software (HP OO) documentation set is made up of the following:

- Help for Central

  Central Help provides information to the following:

  - Finding and running flows
  - For HP OO administrators, configuring the functioning of HP OO
  - Generating and viewing the information available from the outcomes of flow runs

  The Central Help system is also available as a PDF document in the HP OO home directory, in the \Central\docs subdirectory.

- Help for Studio

  Studio Help instructs flow authors at varying levels of programming ability.

  The Studio Help system is also available as a PDF document in the HP OO home directory, in the \Studio\docs subdirectory.

The tutorials are available in the Central and Studio subdirectories of the HP OO home directory.

- Self-documentation for operations and flows in the Accelerator Packs and ITIL folders

  Self-documentation is available in the descriptions of the operations and steps that are included in the flows.

Table of Contents

Objective

The objective of this performance test on Operations Orchestration 9.00 is to obtain the best possible optimal value for product in terms of maximum throughput per hour of flow execution.

Approach

Various combinations of workflows have been designed to replicate the nominal flow execution environment. We have used our own internal tools for execution of flows using backend clients. Four types of workflows are executed with around 10 runs intended to hit the OO server, simulating concurrent users' simultaneous flow execution.

Optimal configuration

This is meant to change few file based configuration options inside product. Tweaking these settings based on the hardware configuration will contribute to more flow throughput/hour. The configuration can be modified as explained below for each Environment

In a Windows environment:

- Maximum concurrent runs default value can be modified in the file "%ICONCLUDE_HOME%\Central\conf\central.properties" by changing the property value "dharma.executor.maxConcurrentRuns"
- Maximum threads default value can be modified in the file "%ICONCLUDE_HOME%\Central\conf\jetty.xml" by changing the property value "maxThreads"
- Maximum heap space default value can be modified in the file "%ICONCLUDE_HOME%\Central\conf\wrapper.conf" by changing the property value "wrapper.java.maxmemory"

In a Linux environment:

- Maximum concurrent runs default value can be modified in the file "$ICONCLUDE_HOME/Central/conf/central.properties" by changing the property value "dharma.executor.maxConcurrentRuns"
- Maximum threads default value can be modified in the file "$ICONCLUDE_HOME/Central/conf/jetty.xml" by changing the property value "maxThreads"
- Maximum heap space default value can be modified in the file "$ICONCLUDE_HOME/Central/conf/wrapper.conf" by changing the property value "wrapper.java.maxmemory"

Tools used

We have used an internal tool for triggering the flow execution on the backend client of Operation Orchestration. No other commercial tools are used for this performance testing.

Name of the internal tool is HeadlessStressTool.jar

System under test environment

Hardware

Below are the different types of hardware configurations used for these tests.

Configuration A

Dell PowerEdge 1950

Intel Xenon Processor

4 CPU

4 GB Memory

Configuration B

Dell PowerEdge 1950

Intel Xenon Processor

8 CPU

8 GB Memory

Configuration C

HP ProLiant DL580 G5

Intel Xenon Processor

16 CPU

32 GB Memory

Configuration D

HP ProLiant DL580 G5

Intel Xenon Processor

16 CPU

64 GB Memory

Configuration E

HP ProLiant DL580 G5

Intel Xenon Processor

16 CPU

8 GB Memory

Operating system

- Windows 2008 Enterprise – OS 1
- Red Hat Enterprise Linux 5.4 – OS2

Flow descriptions

The load is defined as the type of workflow used against Operations Orchestration for running concurrently. We have four groups of batch files used for triggering the flows on each environment. Each batch file corresponds to a group of special flows designed for different levels of flow execution time.

Stress run 1

This run contains flows that can be executed in 2 seconds or less.

Flows used

FastFlowNoRAS

This flow should:

- Run very quickly and never fail.
- Not contain any RAS operations.

Step 1
Execute 'dir' or 'ls' command somewhere on the local Central box.

FastFlowNRAS

This flow should run very quickly and never fail, but it should use an NRAS operation.

Step 1
Use the NRAS Wait operation to wait for 1 second.

FastFlowJRAS

This flow should run very quickly and never fail, but it should use a JRAS operation.

Step 1
Use the JRAS ssh command operation to execute a 'dir' or 'ls' command on the machine on which the JRAS resides.

Stress run 2

This run contains flows that need to perform some actions in the localhost server but that can be done in 10 to 20 seconds.

Flows

BigResultFlowViaJRAS

This flow should:

- Send a lot of data through the context.
- Include at least four steps.
  - One of the steps should send a very large input to the next step through the context.
  - Another step should get back a very large result set, which perhaps can be done by reading a large file. This should be done by sending a large input through the context to a JRAS, and also by getting back a large result set from the JRAS.

Step 1
Ping a box, using JRAS ping, and add a step input that contains about 50K.

Step 2
Take the input from step 1 and write it to a file using the JRAS write-to-file operation.

Step 3
Read in a different file, using a JRAS read-file operation, whose size is 500k, and have it send its results back through Central.

Step 4
Delete the file that was created in step 2.

BigContextFlowNoRAS

Objective: to run a flow that sends large amounts of data through the context. The flow should:

- Include at least four steps.
  - One of the steps should send a very large input to the next step via the context.
  - Another step should get back a very large result set, which perhaps can be done by reading a large file.
- No RASes should be involved in this flow, so everything may need to be done locally on the Central server.

Step 1
Ping a machine, but add a step input that has something like a 50K large input.

Step 2
Take the input from step 1, and write it to a file.

Step 3
Read in a different file, whose size is 500k, and have it send its results back through OO Central.

Step 3
Delete the file that was created in step 2.

Stress Run3

This batch includes the flow that:

- Spawns some multi-level subflows (child flows). A multi-level child flow is a subflow that in turm contains another subflow.
- Waits for completion until all the child flows are executed.

Flows

MegaStepMultiLevelFlow

The flow:

- Executes a couple hundred steps.
- May take a long time or not, but it should at least send some information through the context, and between grandparent, parent, and child flows.
- Should probably make some use of the iterator step:
  - Create three system properties that have at least ten elements that the Iterator step can iterate over.
  - Each of these system properties will be in each of the flows.
  - You can even intersperse RASes in here, if you like. If you do, it could help to stress the RASes more.

Child flow

Step 1
Iterate over the SystemProperty1 elements and send some data through the context.

Parent flow

Step 1
Iterate over the SystemProperty2 elements.

Step 2
For each element in step 1, call child flow. Be sure to send some info into child flow.

Child flow

Step 3
Get back the results from Child flow. You may or may not want to write them to a file.

Grandparent flow

Step 1
Iterate over the SystemProperty3 elements.

Step 2
For each element in step1, call parent flow. Be sure to send some info into parent flow.

ParentFlow

Step 3
Get back the results from parent flow. You may or may not want to write them to a file.

Stress Run4

This contains a flow that sleeps for 2 minutes to complete the execution. This is intended to create flows that execute for a long time, in order to validate the consistency of runs when having the runs are slow to complete.

Flows

TakeAWhileFlow

This flow by definition will run for a long time, which is good for stressing the system, but is not good if you want to see a lot of throughput going through the system. This flow can be as simple as a Wait op (which is an NRAS op), or a command-line sleep call. You could even use both. The point is to get the system to stay busy for a while.

Step 1
Wait operation that waits for 60 seconds.

Step 2
Command line call to 'sleep 60'.

Load choreography

Load choreography is important because it determines how you want to load the system, and takes timing and ryhthm into consideration. In order to do this we can use the Headless Stress Tool (HST). Timing gives the amount of time taken by each flow run or can also be measured by thenumber of runs execute in a particular duration. Whereas rhythm gives an idea of how the runs are varying with time

You may want to use more than one HST in order to set up different load rhythms on Central. For instance, if you included all the above flows in a single runXML, after about 10 minutes, you would notice that all the running flows were TakeAWhileFlows. This is the equivalent of getting a traffic jam because you have a slow poke on a one-lane road. Using multiple HSTs is like opening up more lanes. You might want to divide the flows among the HSTs as follows:

- HST #1: All the fast flows
- HST #2: BigContextNoRAS with MegaStepMultiFlow
- HST #3: TakeAWhileFlow should get its own HST, if you decide to run it
- HST #4: All the other flows

Also, you may need to use more than one instance of an HST, because if you have too many runs and too high a run count, the HST could run out of memory. The number of flows in your runXML times the runcount is the size of the queue. You can either open up a new instance of the HST on the same machine or use a different machine.

Configuring Central to shape its performance

There are some controls that you can tweak in order to make OO Central perform in different ways. Be sure to test the out-of-the-box configurations first as a baseline. After you have done that, you may change some of the following configurations.

Central node

Central.Properties file

```
dharma.headless.librarypool.maxActive = 600
```

Default value: 600

This setting determines the maximum number of headless libraries that you can have at any time. This does not determine the headful libraries that you can have. Headless libraries determine the number of library pools that can be used for headless execution, whereas the headful libraries determine the number of library pools that can be used for headful execution (Execution from UI).

Wrapper.conf

```
wrapper.java.maxmemory=1024
```

Default value: 1024 (Mb of memory)

This setting defines the number of Mb of memory for Central to use as the heap space. We can take this up to 10240 Mb, depending on the RAM available in the machine, but there seems to be a ceiling.

Jetty.xml

```
<Set name="MaxThreads">600</Set>
```

Default value: 600

This setting controls the number of concurrent connections that Jetty can accept. If you go over this limit, you start getting connection reset problems. However, experience shows that increasing this number to the neighborhood of 1500 may not prevent connection reset issues when you have more than 2000 concurrent connections.

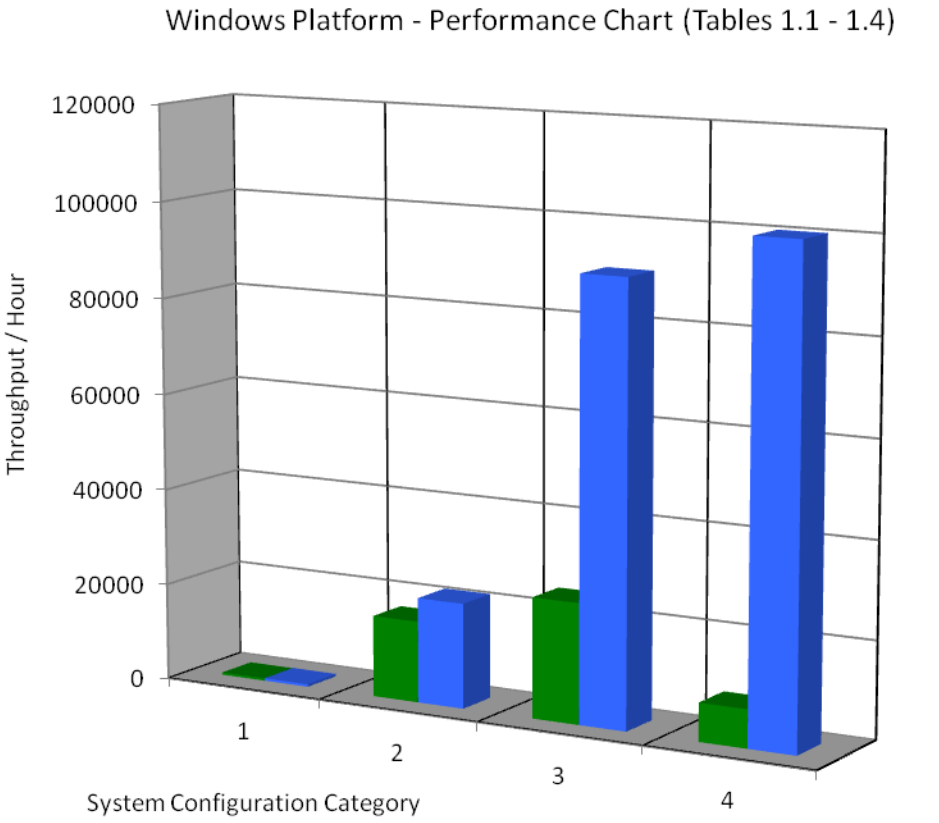
Charts

Single Node OO – Windows Platform

Chart 1, below, depicts the optimal performance observed on the Windows platform with various hardware configurations with optimal values (set of values for above parameters at which the throughput might be high). This chart reflects an installation of OO 9.00 on a single node or a non-clustered central. The System Configurations / OS Categories are as mapped below to the tables in Observed data values for a single node.

- Category 1 – Hardware Config A | OS1 (Table 1.1)
- Category 2 – Hardware Config B | OS1 (Table 1.2)
- Category 3 – Hardware Config C | OS1 (Table 1.3)
- Category 4 – Hardware Config D | OS1 (Table 1.4)

Chart 1



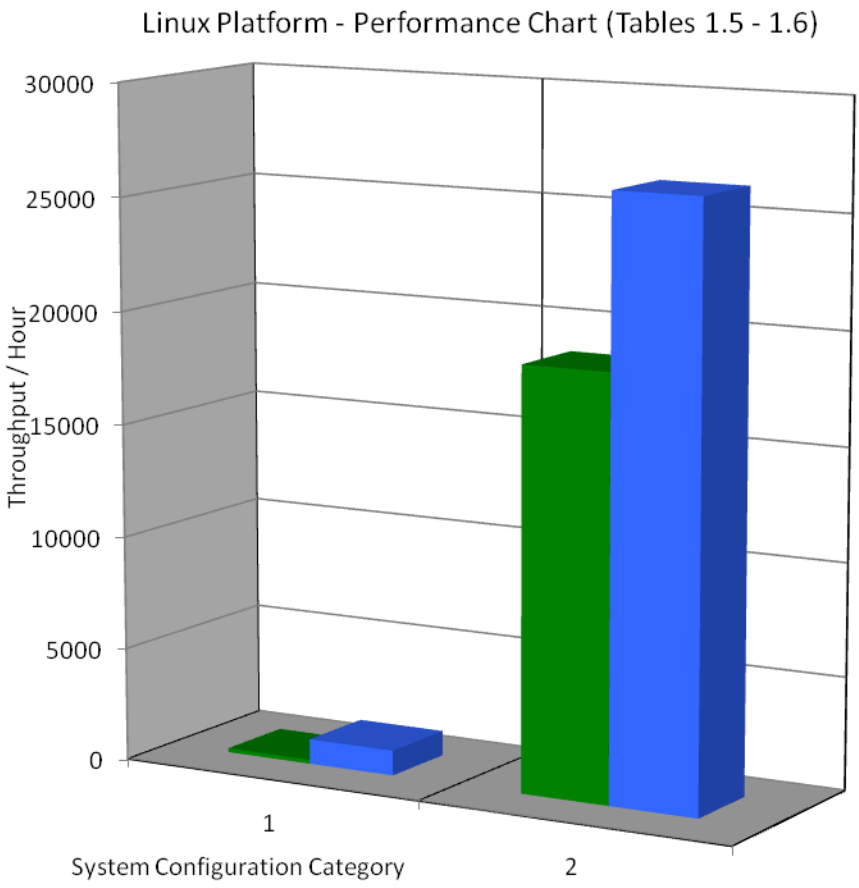
	1	2	3	4
Default	472	16821	24834	7913
Optimal	517	21720	90659	100408

Single Node OO – Linux Platform

Chart 2 depicts the optimal performance observed on Linux platforms on various hardware configurations with optimal values. This chart reflects an installation of OO 9.00 on a single node or a non-clustered Central. The System configuration / OS Category are as mapped below to the tables in Observed data values for a single node..

- Category 5 – Hardware Config E | OS2 (Table 1.5)
- Category 6 – Hardware Config C | OS2 (Table 1.6)

Chart 2



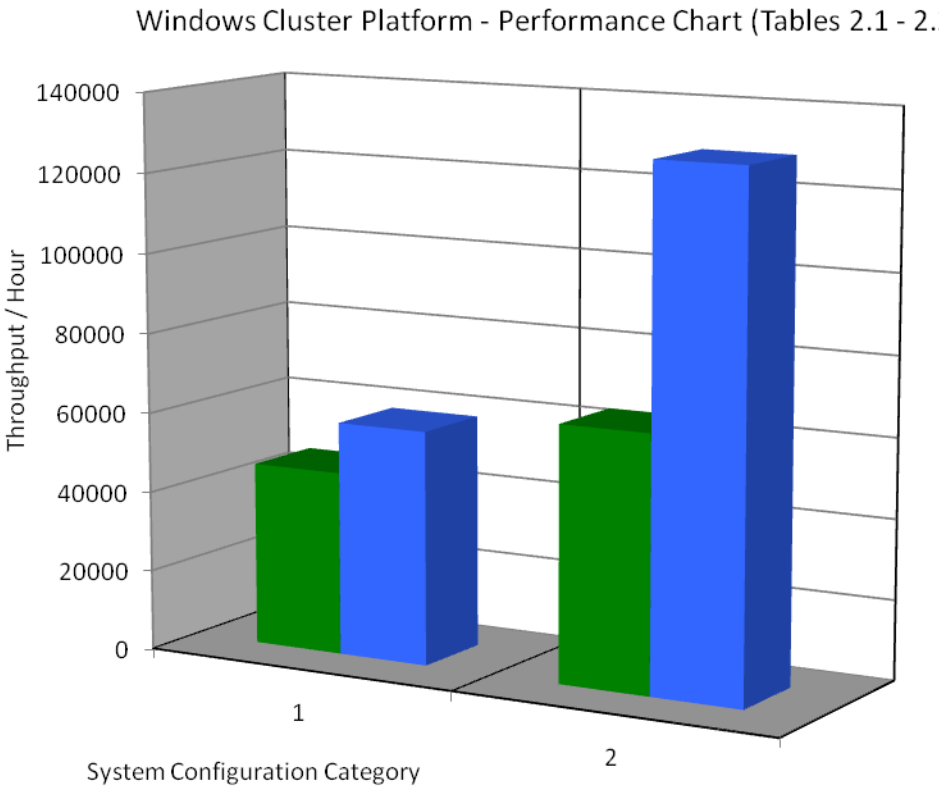
	1	2
Default	197	18666
Optimal	1124	26313

Terracotta Clustered Node OO – Windows Platform

Chart 3 depicts the optimal performance observed on the Windows Platform on various hardware configuration with optimal values. This OO 9.00 setup was done on a Terracotta clustered node with OO load balancer. The System configuration / OS Category are as mapped below to the tables in Observed data values for a Terracotta clustered node.

- Category 1 – Hardware Config E | OS1 (Table 2.1)
- Category 2 – Hardware Config D | OS1 (Table 2.2)

Chart 3



	1	2
Default	45926	64018
Optimal	58620	129052

Observed data values for a single node

Single Node OO – Windows Platform - Category 1 – Hardware Config A | OS1

For single-node Category 1 , we have observed a **9.53%** increase in throughput per hour in the performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 1.1\Chart1)

Table 1.1

Default settings	Throughput /

					Hour
OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	472
	1024	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	60	*2500000*			
Optimal settings					**Throughput / Hour**
OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	517
	1536	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	75	*2500000*			

Single Node OO – Windows Platform - Category 2 – Hardware Config B | OS1

For single-node Category 2, we have observed a **29.12%** increase in throughput per hour in the performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 1.2\Chart1)

Table 1.2

Default settings					**Throughput / Hour**
OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	16821
	1024	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	340	*2500000*			
Optimal settings					**Throughput / Hour**

OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	21720
	2048	*800*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	500	*2500000*			

Single Node OO – Windows Platform - Category 3 – Hardware Config C | OS1

For single-node Category 3, we have observed a **265.06%** increase in throughput per hour in performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 1.3\Chart1)

Table 1.3

Default settings					Throughput / Hour
OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	24834
	1024	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	340	*2500000*			
Optimal settings					**Throughput / Hour**
OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	90659
	3072	*1800*	*1800*	*512*	
Run XML values	Thread Count	Run Count			
	1200	*2500000*			

Single Node OO – Windows Platform - Category 4 – Hardware Config D | OS1

For single-node Category 4 , we have observed a **1168.98%** increase in throughput per hour in the performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 1.4\Chart1)

Table 1.4

Default settings					Throughput / Hour
OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	7913
	1024	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	340	*2500000*			
Optimal settings					**Throughput / Hour**
OO Settings	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	100408
	10240	*2000*	*2000*	*512*	
Run XML values	Thread Count	Run Count			
	1350	*2500000*			

Single Node OO – Linux Platform - Category 5 – Hardware Config E | OS2

For single-node Category 5 , we have observed a **470.56%** increase in throughput per hour in the performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 1.5\Chart2)

Table 1.5

Default settings					Throughput / Hour
	Central Heap Space	Max. concurrent	Max. threads	RAS Heap space	197

		runs			
	1024	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	40	*2500000*			
Optimal settings					**Throughput / Hour**
	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	
	2048	*1500*	*1500*	*512*	1124
Run XML values	Thread Count	Run Count			
	40	*2500000*			

Single Node OO – Linux Platform - Category 6 – Hardware Config C | OS2

For single-node Category 6 , we have observed a **40.97%** increase in throughput per hour in performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 1.6\Chart2).

Table 1.6

Default settings					**Throughput / Hour**
	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	
	1024	*600*	*700*	*512*	18666
Run XML values	Thread Count	Run Count			
	330	*2500000*			
Optimal settings					**Throughput / Hour**
	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	26313

	3072	*2000*	*2000*	*512*	
Run XML values	Thread Count	Run Count			
	330	*2500000*			

Observed data values for a Terracotta clustered node

Terracotta Clustered Node OO – Windows Platform - Category 1 – Hardware Config E | OS1

For clustered-node Category 1 , we have observed a **27.64%** increase in throughput per hour in performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 2.1\Chart3)

Table 2.1

Default settings					Throughput / Hour
	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	45926
	1024	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	75	*2500000*			
Optimal settings					**Throughput / Hour**
	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	58620
	3072	*800*	*800*	*512*	
Run XML values	Thread Count	Run Count			
	75	*2500000*			

Terracotta Clustered Node OO – Windows Platform - Category 2 – Hardware Config D | OS1

For clustered-node Category 2 , we have observed a **101.59%** increase in throughput per hour in performance when the default values of Central heap space, threads and max concurrent runs are tuned for optimal performance (as described in Table 2.2\Chart3)

Table 2.2

Default settings					Throughput / Hour
	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	64018
	1024	*600*	*700*	*512*	
Run XML values	Thread Count	Run Count			
	240	*2500000*			
Optimal settings					**Throughput / Hour**
	Central Heap Space	Max. concurrent runs	Max. threads	RAS Heap space	129052
	10240	*2000*	*2000*	*512*	
Run XML values	Thread Count	Run Count			
	340	*2500000*			

Results

The optimal values that has been identified for each configuration above are the best values observed so far on the test environment. According to these results, the best hardware environment where we were able to achieve 1168% of improvement in performance in Category 4 in a Windows environment.