IBM® Tivoli® Software

IBM Tivoli Dynamic Workload Console V8.6 Capacity Planning

Document version 0.5

Leonardo Lanni
Monica Rossi
Tivoli Workload Automation Performance Team - IBM Rome Lab

IBM
Executive summary

Tivoli Dynamic Workload Console V8.6 capacity plan test activity is composed of a set of 8 test scenarios (divided into 3 categories) that were run against the Tivoli Dynamic Workload Console server on 3 distinct HW configurations with increasing power (in terms of available HW resources), using a specific workload, for a total of 3 combinations.

Test scenarios have the purpose of providing details about the Tivoli Dynamic Workload Console server resource usage while they run, for each of the server configurations.

The following are the 8 scenarios, (grouped into 3 categories), to be run for each configuration:

Monitoring category:

- All jobs in success
- All jobs in error
- All job streams in success
- All job streams in wait

Modeling category:

- Create jobs and job stream
- Modify job

Graphical views category:

- Graphical impact view
- Graphical job stream view

These 8 scenarios were grouped into an appropriate schedule, with 100 users, launched in 60 seconds (a new user launched every 1.67 seconds), each of them running the scenarios in the following way:

Tivoli Dynamic Workload Console V8.6 Capacity Plan schedule:

1. Monitoring (40% of the total number of users)
   - All job streams in wait
   - Wait 2 minutes
   - All job streams in success
   - Wait 2 minutes
   - All jobs in error
   - Wait 2 minutes
   - All jobs in success

2. Modeling (20% of the total number of users)
   - Create jobs and job stream
IBM Tivoli Dynamic Workload Console V8.6 Capacity Planning

- Wait 2 minutes
- Modify job

3. Graphical views (40% of the total number of users)
   - Graphical impact view
   - Wait 4 minutes
   - Graphical job stream view

Tests were run in a completely virtualized environment, based on a P7 server machine with three AIX 7.1 lpar systems to host Tivoli Workload Scheduler V8.6 database, Tivoli Workload Scheduler V8.6 engine, and the Tivoli Dynamic Workload Console V8.6 as the test object.

Monitoring tools were used to record the Tivoli Dynamic Workload Console V8.6 server resource usage, for each test run.

Collected data was analyzed and developed, to provide the reader with information about correct configurations, settings, and tunings to obtain suitable performances for each test.

Test scenario descriptions, analysis, results, and recommendations are given in this document, to guide the reader in the process of building a suitable scheduling environment for their own input workload.

Each test was successfully completed on each of the 3 Tivoli Dynamic Workload Console V8.6 server configurations, either from a functional point of view or from a performance/scalability point of view reaching the following result for the upper scaling configurations:

1. Tivoli Dynamic Workload Console V8.6 Capacity plan schedule: up to 300 users were able to successfully complete the whole schedule from a functional point of view, with a workload of 100,000 jobs and 200,000 dependencies, but not acceptable performances.

2. Tivoli Dynamic Workload Console V8.6 Capacity plan schedule: up to 200/250 users were able to successfully complete the whole schedule from a functional point of view and could also grant scalability and good/acceptable performances.

Collected test results, in terms of monitored Tivoli Dynamic Workload Console V8.6 Capacity server resource usage, were analyzed and developed to provide useful information to get the best performances for the scheduling environment to be built.

In addition, for 100 users, the schedule has been run changing the Tivoli Workload Scheduler V8.6 embedded WebSphere Application Server (eWAS) V7.0.0.0 (FP15) garbage collector policy, from the default one (optthruput) to gencon and subpool.

Test results highlighted the fact that changing the garbage collector policy does not give any improvement, in terms of performance, so it is possible to assert that the best tuning for the garbage policy collector is to use the default one.

The following table offers a summary of the obtained results and highlights the best configurations for the Tivoli Dynamic Workload Console V8.6, to execute a workload similar to the one used in this test activity.
<table>
<thead>
<tr>
<th>Number of users</th>
<th>Test</th>
<th>Best configuration for performance</th>
<th>Best configuration for performance/cost</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Capacity Planning</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Good performances with best performance configuration. Acceptable performances with best performance/cost configuration;</td>
</tr>
<tr>
<td>100</td>
<td>Garbage Collector tuning</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Use the default opttrhuput policy</td>
</tr>
<tr>
<td>100</td>
<td>Scalability/ Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Good results with best performance configuration. Acceptable results with best performance/cost configuration.</td>
</tr>
<tr>
<td>200</td>
<td>Scalability/ Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Acceptable results for both the configurations.</td>
</tr>
<tr>
<td>250</td>
<td>Scalability/ Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Acceptable results with best performance configuration. Tolerable results with best performance/cost configuration.</td>
</tr>
<tr>
<td>300</td>
<td>Scalability/ Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Only functionality is granted for both the configurations.</td>
</tr>
</tbody>
</table>

Table 1 - Test activities results
Executive summary .................................................................................................................. 3

Contents ................................................................................................................................. vi

List of Figures .......................................................................................................................... viii

List of Tables ............................................................................................................................ ix

Revision History ...................................................................................................................... x

1 Goal ........................................................................................................................................ 1

2 Approach ............................................................................................................................... 1

   2.1 Installation and setup ........................................................................................................ 1

   2.2 Test initial conditions ....................................................................................................... 2

   2.3 Database information and tuning settings ....................................................................... 2

   2.4 Rational Performance Tester V8.2 .................................................................................. 3

3 Environment .......................................................................................................................... 4

   3.1 Tivoli Workload Scheduler V8.6 engine .......................................................................... 5

       3.1.1 Tivoli Workload Scheduler V8.6 engine scheduling objects ................................. 6

   3.2 Tivoli Dynamic Workload Console V8.6 server .............................................................. 6

       3.2.1 Tivoli Dynamic Workload Console V8.6 embedded WebSphere Application

           Server heap size configuration and tuning ................................................................. 7

       3.2.2 Tivoli Dynamic Workload Console V8.6 embedded WebSphere Application

           Server Java process garbage collector policy configuration and tuning ................... 8

   3.3 DB2 Database .................................................................................................................. 9
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Network</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Scenarios</td>
<td>10</td>
</tr>
<tr>
<td>4.1</td>
<td>Monitoring test scenarios</td>
<td>11</td>
</tr>
<tr>
<td>4.2</td>
<td>Modeling test scenarios</td>
<td>12</td>
</tr>
<tr>
<td>4.3</td>
<td>Graphical view test scenarios</td>
<td>13</td>
</tr>
<tr>
<td>4.4</td>
<td>Test pre-conditions and ramp-up schedule</td>
<td>15</td>
</tr>
<tr>
<td>4.5</td>
<td>Capacity plan concurrency schedule</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Results</td>
<td>18</td>
</tr>
<tr>
<td>5.1</td>
<td>Capacity Plan</td>
<td>18</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Conclusions and recommendations</td>
<td>24</td>
</tr>
<tr>
<td>5.2</td>
<td>Breaking point</td>
<td>24</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Conclusions and recommendations</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>Garbage collector policies comparison</td>
<td>28</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Conclusions and recommendations</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>29</td>
</tr>
</tbody>
</table>
Figure 1 - DB2 tunings ........................................................................................................ 10
Figure 2 - Graphical Impact View ....................................................................................... 14
Figure 3 - Graphical Job Stream View .................................................................................. 14
Figure 4 - Tivoli Dynamic Workload Scheduler V8.6 Capacity Plan ramp-up preliminary schedule 16
Figure 5 - Tivoli Dynamic Workload Scheduler V8.6 Capacity Plan concurrency schedule - details ........................................................................................................ 17
Figure 6 - All job streams in wait response time - comparison for all the configurations ........... 21
Figure 7 - Graphical impact view response time - comparison for all the configurations .......... 22
Figure 8 - All job streams in success response time for different concurrent users, all configurations comparison ........................................................................................................ 25
Figure 9 – Job streams view response time for different concurrent users, all configurations comparison ........................................................................................................ 26
LIST OF TABLES

Table 1 - Test activities results ................................................................. 5
Table 2 - Rational Performance Tester V8.2 environment description .................. 4
Table 3 - Tivoli Dynamic Workload Console V8.6 Capacity Plan test environment ........ 5
Table 4 - Tivoli Dynamic Workload Console V8.6 configuration descriptions .............. 7
Table 5 - Heap size initial and maximum values for each Tivoli Dynamic Workload Console V8.6 server configuration ........................................................... 8
Table 6 - Garbage collector policy settings into server.xml configuration file ............... 9
Table 7 - Database server machine details ................................................................ 9
Table 8 - Monitoring category - main steps and result set details .............................. 12
Table 9 - Modeling category - Steps details .......................................................... 13
Table 10 - Graphical view category - steps details ............................................... 14
Table 11 - Capacity plan - concurrency schedule test scenarios main steps .................. 20
Table 12 - Capacity Plan results ........................................................................ 24
Table 13 - Test activities results ........................................................................... 30
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revised By</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>15/11/11</td>
<td>0.1</td>
<td>M. Rossi</td>
<td></td>
</tr>
<tr>
<td>15/11/11</td>
<td>0.2</td>
<td>L. Lanni</td>
<td></td>
</tr>
<tr>
<td>15/11/11</td>
<td>0.3</td>
<td>M. Rossi</td>
<td></td>
</tr>
<tr>
<td>25/11/11</td>
<td>0.4</td>
<td>L. Lanni</td>
<td></td>
</tr>
<tr>
<td>28/11/11</td>
<td>0.5</td>
<td>M. Rossi</td>
<td></td>
</tr>
</tbody>
</table>
1 Goal

The purpose of the Tivoli Dynamic Workload Console V8.6 Capacity Planning test activity is to provide information and answers to a customer that needs to identify and size a suitable environment to successfully manage its workload with a specific number of users. By reading this document, a customer should be able to decide, configure, and set up an environment, in terms of hardware servers provided with appropriate features, and network infrastructure, so that the workload, in terms of users and scheduling objects, can be successfully managed by the Tivoli Dynamic Workload Console V8.6 server.

Another goal is to identify, for the given workload, in terms of number of concurrent users, considering also the test scenarios and the scheduling objects, the scalability of the Tivoli Dynamic Workload Console V8.6, that is the maximum number of users it is able to successfully serve and manage, from a functional point of view and then from a performance point of view.

A third goal is to evaluate, for a fixed number of concurrent users, scheduling objects and test scenarios, which is the best Tivoli Dynamic Workload Console V8.6 embedded WebSphere Application Server V7.0.0.0 (FP15) Java process garbage collector policy, if any, to get the best performances in terms of response times.

Tivoli Dynamic Workload Console V8.6 was installed on a server machine provided with the possibility of changing the hardware configuration. The Tivoli Dynamic Workload Console V8.6 server machine was built on a P7 Server machine with the capability of hosting virtual machines with the required features, and the possibility of changing the configuration of a virtual machine, modifying the number of core processors and the RAM available.

Although the server on which Tivoli Dynamic Workload Console V8.6 was installed is located on a virtual machine, the virtualization environment means that resources for the server are dedicated and not shared, so that performance cannot be affected by any other virtual machine hosted by the same P7 virtualization infrastructure.

In this way, it was possible to run the given workload on 3 distinct Tivoli Dynamic Workload Console V8.6 server hardware configurations, without the need for switching on a distinct physical machine provided with different hardware features.

Tests were run with a single predefined input workload. Further details about this workload are available in the corresponding “Workload” section.

2 Approach

2.1 Installation and setup

The first preliminary phase of the Tivoli Dynamic Workload Console V8.6 Capacity Planning test activity was the installation and setup of the entire environment.

To obtain a full working environment for the purpose of the capacity planning activities, the following components were installed:
Approach

- Tivoli Workload Scheduler V8.6 engine (Global Availability code level), installed following the normal graphical installation, in the default path, on a dedicated server machine, starting from a scratch server clean configuration.

- DB2 V9.7.0.3 Database (DB2), installed following the normal graphical installation, in the default path, on a dedicated distinct server machine, starting from a scratch server clean configuration.

- Tivoli Dynamic Workload Console V8.6 (Global Availability code level), installed following the normal graphical installation, in the default path, on a dedicated distinct server machine, starting from a scratch server clean configuration.

All the components were installed in clean environments, that is on scratch server machines dedicated to host only the components described above, so that no other significant application (in terms of resource consumption) run on the same machines during the test process.

Where possible, snapshots of virtualized machines and databases were taken after the components were successfully installed to make it possible to easily revert to a safe initial clean configuration at any new test start, restoring the exact initial conditions for each test run during the capacity plan activity.

2.2 Test initial conditions

To perform test activities in the same conditions for all runs, so that the only difference between the current run and the previous one is an intentionally modified feature (for example configuration or workload), the following setup protocol was followed, where applicable:

- Stop the current applications

- Clean all the traces, logs, and written data on the application server machine and store them if needed

- Check that machine health status is OK (free memory, RAM, CPU, disk space, and so on)

- Reboot the system machine, if needed

- Restart the applications to obtain a clean scratch environment.

It is useful to note that all the Tivoli Workload Dynamic Console V8.6 Capacity Plan test activities were run with the scheduling feature disabled because the target of the test activity is the web user interface of Tivoli Workload Scheduler V8.6.

2.3 Database information and tuning settings

All the tests were run using a dedicated database server machine. The database application was IBM DB2 Version 9.7.0.3, in a 64-bit environment.

Database DB2 was configured with the following tuning modifications, to improve and optimize performances when used with Tivoli Workload Scheduler V8.6.
For Tivoli Workload Scheduler DB:

Log file size (4KB) \((\text{LOGFILSZ}) = 10000\)

Number of primary log files \((\text{LOGPRIMARY}) = 80\)

Number of secondary log files \((\text{LOGSECOND}) = 40\)

For embedded WebSphere Application Server:

\[
\text{<connectionPool xmi:id="ConnectionPool_1174319908875" connectionTimeout="180" maxConnections="50" minConnections="1" reapTime="180" unusedTimeout="1800" agedTimeout="0" purgePolicy="EntirePool" numberOfSharedPoolPartitions="0" numberOfUnsharedPoolPartitions="0" numberOfFreePoolPartitions="0" freePoolDistributionTableSize="0" surgeThreshold="-1" surgeCreationInterval="0" testConnectionInterval="15" testConnection="true" stuckTimerTime="0" stuckTime="0" stuckThreshold="0"/>}
\]

The database component was not changed, in terms of configuration or any other modifiable parameter, during the entire test activity for capacity plan; some database server machine reboots were sometimes made, to ensure that the server was always healthy with no configuration/hardware problems and issues.

The only small differences in the database, between one particular test case and another, could only have been the particular workload, in terms of scheduling objects present in the database itself.

2.4 Rational Performance Tester V8.2

As mentioned before, the Capacity Plan test activities for Tivoli Dynamic Workload Console V8.6 are based on the concurrency of 100 users that perform a set of 8 test scenarios divided into 3 main categories (monitoring, modeling, and graphical views).

To reproduce the concurrency of such a big number of concurrent users a suitable application was used: Rational Performance Tester V8.2.

Rational Performance Tester V8.2 allows you to record and play back test scenarios against any web user interface of a certain application, like the Tivoli Dynamic Workload Console V8.6.

After the test scenarios were recorded, processed and successfully played back, it is possible to group them into a "schedule" that simulates a certain number of concurrent users.

In more detail, a schedule is a sort of test scenarios container that allows you to decide how the users will execute which test scenarios, in terms of percentage of users executing certain test scenarios (gathering them into groups), and time constraints (for example, adding delays so that a user will execute a certain test scenario, wait for a time interval, and finally execute another test scenario).
Rational Performance Tester V8.2 is composed of one controller server and one or more agents.

The controller is responsible for hosting the application to record and playback test scenarios, and organize them into schedules.

The agents are responsible for running the test run against the target web user interface; in other words, the agent is the component that is in charge of providing requests to the test target application, collecting responses, and simulating the recorded user interaction in a fully automated way, with the possibility of inserting verification points to check that the actual server response is the same as the one obtained at recording time.

Controller can act also as an agent, but it is possible to install one or more agents on distinct server machines to avoid saturating the controller resources, leading to a test or schedule run failure.

In the Tivoli Dynamic Workload Console V8.6 test activities, Rational Performance Tester V8.2 was used in the following configuration:

<table>
<thead>
<tr>
<th>Role</th>
<th>OS</th>
<th>CPU</th>
<th>RAM</th>
<th>HDD</th>
<th>32/64 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rational Performance Tester V8.2 Controller</td>
<td>Windows Server 2008 EE x64</td>
<td>2 x Dual Core Intel Xeon 1.86 GHz</td>
<td>12 GB</td>
<td>2 x Hitachi HDS721616PLA (SATA, 160 GB, 7200 rpm) – RAID 1</td>
<td>64</td>
</tr>
<tr>
<td>Rational Performance Tester V8.2 Agent (4 instances)</td>
<td>Rhel 5.1</td>
<td>2 x Dual Core Intel Xeon 1.60 GHz</td>
<td>8 GB</td>
<td>HD1:140 GB - HD2:140 GB</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 2 - Rational Performance Tester V8.2 environment description

As can be seen from the table, the Rational Performance Tester V8.2 was installed on 2 servers; the first one hosting the controller, while a suitable dedicated server machine was used to host the agent component, to avoid bottlenecks in the workload and concurrency generation during the test activities.

The agent component server hosts 4 instances of the Rational Performance Tester V8.2 Agent, so that the server can be fully used and leveraged.

3 Environment

The Tivoli Dynamic Workload Console V8.6 Capacity Plan environment, with the purpose of collecting data and information about Tivoli Dynamic Workload Console V8.6 server machine performance and resource consumption, is composed of three components: Tivoli Workload Scheduler V8.6 engine with its own database DB2 V9.7.0.3 and Tivoli Dynamic Workload Console V8.6 server.
The following picture displays the entire capacity plan test environment:

3.1 Tivoli Workload Scheduler V8.6 engine

The target of this test is the Tivoli Dynamic Workload Console V8.6 server. For this reason, the other components, Tivoli Workload Scheduler V8.6 engine and database DB2 V9.7.0.3, were installed on a predefined fixed hardware configuration:

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPU</th>
<th>RAM</th>
<th>SWAP</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tivoli Workload Scheduler V8.6 engine</td>
<td>8 x 4 PowerPC_POWER7 3300 MHz</td>
<td>16 GB</td>
<td>2 GB</td>
<td>120 GB</td>
</tr>
</tbody>
</table>

Processor type is a Power7 with a clock frequency of 3300 MHz and 4 logical cores for each of the 8 physical cores. Swap paging space is set to 2 GB, and the hard drive has 120 GB of space.
Heap size for embedded WebSphere Application Server V7.0.0.0 (FP15) was configured with an initial heap size of 1 GB, and a maximum heap size of 4 GB.

### 3.1.1 Tivoli Workload Scheduler V8.6 engine scheduling objects

Tivoli Workload Scheduler V8.6 database was populated with the following scheduling objects:

- 100,000 jobs
- 5,000 job streams
- 200,000 dependencies (approximately)

The jobs were grouped into 5,000 job streams and each job stream contains 20 jobs.

Dependencies are composed of the following types:

- Job “follows” job dependencies: 150,000, 30 for each job stream, only between jobs belonging to the same job stream.
- Job streams “follows” job stream dependencies: 3,333, each job stream in the number sequence \(1, 4, 7, 3i + 1\), with \(i \geq 0\) has 2 “follows” dependencies towards the 2 next job streams.
- Prompt dependencies: 15,000, 3 jobs in each job stream have a prompt dependencies from a set of created prompts.
- Resource dependencies: 15,000, 3 jobs in each job stream need resources from a set of created resources.
- File dependencies: 15000, 3 jobs in each job stream open files from a set of 4 files that were opportunely created on the engine server.

The number of dependencies is 198,333.

### 3.2 Tivoli Dynamic Workload Console V8.6 server

The Tivoli Dynamic Workload Console V8.6 server is implemented by a unique server, hosted by an AIX 7.1.0.0 System (64-bit architecture), virtualized over a P7 server environment.

The Tivoli Dynamic Workload Console V8.6 hardware resources are dedicated, so that they cannot be modified, influenced, or changed by any other system hosted by the same P7 server environment.

The virtualization allows the Tivoli Dynamic Workload Console V8.6 server configuration to be dynamically changed, in terms of amount of cores and RAM available, leading to a total number of 3 different engine configurations, detailed in the table below:
Table 4 - Tivoli Dynamic Workload Console V8.6 configuration descriptions

<table>
<thead>
<tr>
<th>Config</th>
<th>CPU</th>
<th>RAM</th>
<th>SWAP</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2 x 4 PowerPC_POWER7 3300 MHz</td>
<td>8 GB</td>
<td>2 GB</td>
<td>120 GB</td>
</tr>
<tr>
<td>C2</td>
<td>4 x 4 PowerPC_POWER7 3300 MHz</td>
<td>8 GB</td>
<td>2 GB</td>
<td>120 GB</td>
</tr>
<tr>
<td>C3</td>
<td>8 x 4 PowerPC_POWER7 3300 MHz</td>
<td>8 GB</td>
<td>2 GB</td>
<td>120 GB</td>
</tr>
<tr>
<td>C4</td>
<td>8 x 4 PowerPC_POWER7 3300 MHz</td>
<td>16 GB</td>
<td>2 GB</td>
<td>120 GB</td>
</tr>
</tbody>
</table>

Configurations will be referred to, in this document, as C1, C2, C3, and C4; Config1, Config2, Config3 and Config4; Cfg1, Cfg2, Cfg3, and Cfg4. In each case, they refer to the same configurations specified in Table 4.

Each configuration is provided with a fixed value of 2 GB of swap paging space, and of a hard disk of 120 GB.

The variable features, configuration by configuration, are the number of physical processors (2, 4, and 8) and the amount of RAM (8 GB or 16 GB).

Processor type is always the same: a Power7 with a clock frequency of 3300 MHz and equipped with 4 logical cores for each physical unit.

Configurations are ordered by number of processors and, then, by amount of RAM. This gives a sort of power order for the configurations, so that C1 is the weakest configuration, while C4 is the most powerful configuration.

The Tivoli Dynamic Workload Console code version installed on this machine is the 8.6 GA (Global Availability); Tivoli Dynamic Workload Console has also the embedded WebSphere Application Server (eWAS) V7.0.0.0 (FP15) installed, the server is built over a 64-bit Java Virtual Machine with the following features: IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 AIX).

### 3.2.1 Tivoli Dynamic Workload Console V8.6 embedded WebSphere Application Server heap size configuration and tuning

An important tuning is related to the heap size of the embedded WebSphere Application Server Java process the Tivoli Dynamic Workload Console V8.6 is based on. This parameter is used to configure the amount of primary RAM memory that the embedded WebSphere Application Server can use to allocate the objects that it needs to manage during normal working activities.

It is possible to set the initial size of the heap memory, that is, the amount of memory that the embedded WebSphere Application Server Java process takes for itself, and the maximum size of heap memory, that is, the maximum amount of memory the embedded WebSphere Application Server Java process can get, whenever needed.

It is important to remember that the amount of heap size, both for initial and maximum values, is upper-bounded by the available RAM of the server hosting the application.

On the other side, it is a well-known performance best practice not to reserve the whole system RAM available for the Java embedded WebSphere Application Server process, but only a reasonable percentage, so that other system processes, necessary for a healthy system status, can run correctly with no resource availability issues.
Considering these factors, this rule was used to determine the initial and maximum heap size for each tested hardware configuration:

- Maximum heap size: 50% of the total physical RAM available
- Initial heap size: 50% of the maximum heap size

This formula led to the following heap values, for all the 4 configurations, summarized in the following table:

<table>
<thead>
<tr>
<th>Config</th>
<th>RAM</th>
<th>initial heap size</th>
<th>maximum heap size</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>8.0 GB</td>
<td>2.0 GB</td>
<td>4.0 GB</td>
</tr>
<tr>
<td>C2</td>
<td>8.0 GB</td>
<td>2.0 GB</td>
<td>4.0 GB</td>
</tr>
<tr>
<td>C3</td>
<td>8.0 GB</td>
<td>2.0 GB</td>
<td>4.0 GB</td>
</tr>
<tr>
<td>C4</td>
<td>16.0 GB</td>
<td>4.0 GB</td>
<td>8.0 GB</td>
</tr>
</tbody>
</table>

Table 5 - Heap size initial and maximum values for each Tivoli Dynamic Workload Console V8.6 server configuration

### 3.2.2 Tivoli Dynamic Workload Console V8.6 embedded WebSphere Application Server Java process garbage collector policy configuration and tuning

Part of the test activity is the garbage collection policy tuning, whose goal is to find the most performing policy (if any), in terms of minimum response times.

To perform this activity, the concurrency schedule was run, on each of the first 3 configurations C1, C2, C3, setting the garbage collection policy to:

1. optthruput
2. gencon
3. subpool

To change the garbage collector policy, the `server.xml` configuration file of the embedded WebSphere Application Server V7.0.0.0 (FP15) of the Tivoli Dynamic Workload Console V8.6 was edited and changed according to the following table.
3.3 DB2 Database

The Tivoli Workload Scheduler V8.6 database used for this capacity plan test is IBM DB2 V9.7.0.3 for AIX. The database is hosted by a distinct AIX 7.1.0.0 (64 bit architecture) server machine, hosted by the P7 virtualization environment, with the following features:

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPU</th>
<th>RAM</th>
<th>SWAP</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>4 x 4 PowerPC POWER7 3300 MHz</td>
<td>8 GB</td>
<td>2 GB</td>
<td>120 GB</td>
</tr>
</tbody>
</table>

The database server machine is provided with 4 physical Power7 processors with clock frequency of 3,300 MHz, 8 GB of RAM and a swap space of 2 GB. The server is provided with a 120 GB hard drive.

The following figure displays the database configuration tuning that was performed on the scheduling environment.

The first section of the figure refers to modifications that need to be performed directly on the DB2 database, and can be obtained executing on the database server machine, under the database owner user, the following command:

```
  db2 update db cfg for TWS86 using LOGPRIMARY 80
  db2 update db cfg for TWS86 using LOGSECOND 40
  db2 update db cfg for TWS86 using LOGFILSIZ 10000
```

The second section of the figure refers to modifications that need to be performed against the “resources.xml” file that is present on the embedded WebSphere Application Server V7.0 (FP15) on which the Tivoli Workload Scheduler V8.6 Engine is installed.
3.4 Network

The Tivoli Workload Scheduler V8.6 engine, Tivoli Dynamic Workload Console V8.6 server, DB2 V9.7.0.3 and the Rational Performance Tester V8.2 environment (Controller and Agent) communicate over a 1 Gbit Ethernet network and adapters.

This means that, under normal working conditions of the network infrastructure, no bottleneck is expected from the links connecting the distinct components involved in these test activities, because the bandwidth is large enough to allow heavy traffic and data exchanges.

4 Scenarios

Tivoli Dynamic Workload Console V8.6 server capacity plan testing consists of a sequence of scenarios, grouped inside an appropriate Rational Performance Tester schedule to be run, and to collect data about the hardware resource usage, performance, and scalability results obtained.

Two main activities were run, for each of the 4 configurations for Tivoli Dynamic Workload Console V8.6 server machine:

- Capacity plan: for 100 concurrent users, run the concurrency schedule
- Breaking point discovery: increase the number of concurrent users to determine the breaking point (that is maximum number of users the server is able to manage, in terms of functional accuracy and performance).
A conclusive third activity performed, was the embedded WebSphere Application Server V7.0.0.0 (FP15) garbage collection policies test (changed from the default one (opttrhu), to the subpool and gencon policies).

All the tests run the same concurrency schedule that is composed of test scenarios:

- Monitoring test scenarios (4 test scenarios)
- Modeling test scenarios (2 test scenarios)
- Graphical view test scenarios (2 test scenarios)

Each test scenario is composed of a sequence of steps that a user would perform during the interaction with the web user interface, to complete a task successfully.

For each of these test scenarios, the focus was on:

- The most significant steps, correlated to the task (for example the query request and result set display for a monitoring test scenario)
- Any step that required a higher time than expected, whenever a threshold of maximum tolerable response time was exceeded (5 seconds or 10 seconds, depending on the number of concurrent users)
- The login step.

All the scenarios are repeated for all the 4 Tivoli Dynamic Workload Console V8.6 server configurations.

Each test scenario was run using a user selected from the users’ data pool set that contains 250 users, numbered from pvt0 to pvt249. Each user has two roles assigned: TWSWEBUIAdministrator and TEDWBAAdministrator.

### 4.1 Monitoring test scenarios

The monitoring category contains 4 test scenarios. Each scenario represents a typical way of monitoring the jobs and job streams status by running appropriate tasks to retrieve scheduling objects matching the required filter criteria.

These are the 4 test scenarios with their detailed steps:

1. All job streams in wait:
   - Log in to Tivoli Dynamic Workload Console V8.6
   - Go to TWS => Monitor => Workload => Monitor Job Streams
   - Launch “All Job Streams in WAIT” task
   - Find a particular job stream
   - Log out

2. All job streams in success:
   - Log in to Tivoli Dynamic Workload Console V8.6
   - Go to TWS => Monitor => Workload => Monitor Job Streams
   - Launch “All Job Streams in SUCC” task
Scenarios

- Select the first returned job stream and click the button to display dependencies
- Log out

3. All jobs in success:
   - Log in to Tivoli Dynamic Workload Console V8.6
   - Go to TWS => Monitor => Workload => Monitor Jobs
   - Perform “All Jobs in SUCC” task
   - Select a particular job and display the related job log
   - Log out

4. All jobs in error:
   - Log in to Tivoli Dynamic Workload Console V8.6
   - Go to TWS => Monitor => Workload => Monitor Jobs
   - Perform “All Jobs in ERR” task
   - Find a particular job
   - Log out

The following table summarizes the size of the result set for each test scenario, and highlights the particular specific step performed on an item of the result set.

The other significant steps were selected by picking the steps, from each test scenario, that have a response time higher than a tolerable threshold (from 5 to 10 seconds, depending on the number of concurrent users).

<table>
<thead>
<tr>
<th>Test scenario</th>
<th>Category</th>
<th>Main step</th>
<th>Other significant steps</th>
<th>Result set</th>
</tr>
</thead>
<tbody>
<tr>
<td>All job streams in success</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Login, monitor job streams, dependencies</td>
<td>400</td>
</tr>
<tr>
<td>All job streams in wait</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Login, monitor job streams</td>
<td>100</td>
</tr>
<tr>
<td>All jobs in success</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Login</td>
<td>10000</td>
</tr>
<tr>
<td>All jobs in error</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Login</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 8 - Monitoring category - main steps and result set details

4.2 Modeling test scenarios

The modeling category contains 2 test scenarios, describing another typical use of the Tivoli Dynamic Workload Console V8.6, which is the creation or the editing of scheduling objects like jobs and job streams:

1. Create jobs and job stream
   - Log in to Tivoli Dynamic Workload Console V8.6
   - Create Workload definitions
   - Select the engine
   - Create a new job (0)
   - Edit, save, and close the job (0)
   - Create a new job (1)
   - Edit, save, and close the job (1)
   - Create new job stream
- Add job (0) and job (1) to the job stream
- Edit, save, and close the job stream
- Log out

2. Modify job
- Log in to Tivoli Dynamic Workload Console V8.6
- List Workload definitions
- Perform the query to find job
- Edit the job
- Save the job
- Log out

The following table describes the main steps for each scenario that the response time analysis focuses on:

<table>
<thead>
<tr>
<th>Test scenario</th>
<th>Category</th>
<th>Main step</th>
<th>Other significant steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create jobs and job stream</td>
<td>Modeling</td>
<td>Save job stream</td>
<td>Login, create workload definitions, Select engine, edit job</td>
</tr>
<tr>
<td>Modify job</td>
<td>Modeling</td>
<td>perform query</td>
<td>Login, list workload definitions</td>
</tr>
</tbody>
</table>

Table 9 - Modeling category - Steps details

The step choice is based on the response time obtained during the several tests, and on the fact that it is representative of the entire scenario.

4.3 Graphical view test scenarios

The graphical view category contains two test scenarios focused on the graphical views available. The graphical views are visual representations of job stream content and relationships with other job streams.

Each job stream has 2 “follows” dependencies towards other job streams, and contains 20 jobs having 30 “follows” dependencies towards other jobs inside the same job stream.

In addition, jobs inside each job stream have 3 prompt dependencies, 3 resource dependencies, and 3 file dependencies.

1. Graphical Impact View
   - Log in to Tivoli Dynamic Workload Console V8.6
   - Go to TWS => Monitor => Workload => Monitor Job Streams
   - Perform the graphical task to get a particular job stream
   - Select the returned job streams => graphical view => impact view
   - Close the graphical impact view window
   - Log out

2. Graphical Job stream View
   - Log in to Tivoli Dynamic Workload Console V8.6
   - Go to TWS => Monitor => Workload => Monitor Job Streams
   - Perform the graphical task to get a particular job stream
   - Select the returned job streams => graphical view => job stream view
   - Close the graphical impact view window
   - Log out
The main steps are the graphical rendering when the new window for the graphical view is opened, and the query performed to get the specific job stream on which the graphical view is performed.

The following table summarizes this information:

<table>
<thead>
<tr>
<th>Test scenario</th>
<th>Category</th>
<th>Main step</th>
<th>Other significant steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphical Impact view</td>
<td>Graphical view</td>
<td>Graphical Impact view</td>
<td>Login, graphical task</td>
</tr>
<tr>
<td>Graphical Job stream</td>
<td>Graphical view</td>
<td>Graphical Job stream view</td>
<td>Login, graphical task</td>
</tr>
</tbody>
</table>

*Table 10 - Graphical view category - steps details*

The following figures display information about the two graphical view test scenarios, in terms of scheduling object involved in the rendering.

*Figure 2 - Graphical Impact View*

*Figure 3 - Graphical Job Stream View*
4.4 Test pre-conditions and ramp-up schedule

Before running the test scenarios in the schedule, a series of steps are performed to run the tests always in the same initial conditions:

- Check if the database server is up and running and healthy using tools like topas; reboot the database server and restart the database if needed

- Check if the engine server is up and running and healthy using tools like topas; if needed, reboot the server, clean and restart the whole Tivoli Workload Scheduler V8.6 engine environment; start nmon tool to monitor system performance (useful to find any bottlenecks in the subsequent analysis)

- Check if the Tivoli Dynamic Workload Console V8.6 server is up and running and healthy using topas; reboot the server if necessary

- On the Tivoli Dynamic Workload Console V8.6 server, clean embedded WebSphere Application Server V7.0.0.0 (FP15) logs and traces

- On the Tivoli Dynamic Workload Console V8.6 server, start nmon

- On the Tivoli Dynamic Workload Console V8.6 server, look at the SystemOut.log and SystemErr.log to check that no exception occurs during the test

After the sequence of these preliminary steps is completed, a preliminary ramp-up schedule is needed, before the concurrency schedule runs, to make the Tivoli Dynamic Workload Console V8.6 server able to initialize all the components.

This is needed to preserve the response time measurements to be affected by the initialization times, higher than the average response times.

The ramp-up schedule is composed of 8 users (launched one every 10 seconds) performing one of the 8 test scenarios composing the schedule of the Tivoli Dynamic Workload Console V8.6 server Capacity Plan test activities.

The following screen capture displays this schedule, when it is launched under Rational Performance Tester V8.2 Controller:
Figure 4 - Tivoli Dynamic Workload Scheduler V8.6 Capacity Plan ramp-up preliminary schedule
4.5 Capacity plan concurrency schedule

The concurrency schedule contains all the 8 test scenarios previously described, organized in an appropriate way, so that users will run them, concurrently, with some time constraints (delays) and percentage of users running particular test scenarios constraints.

This is the concurrency schedule, launched using Rational Performance Tester V8.2 Controller:

- All 100 users launched in 60 seconds (a new user each 0.6 seconds)
  - Monitoring (run by the 40% of the users)
    - All job streams in wait
    - Wait 2 minutes
    - All job streams in succ
    - Wait 2 minutes
    - All jobs in error
    - Wait 2 minutes
    - All jobs in success
  
  - Modeling (run by the 20% of the users)
    - Create jobs and job stream
    - Wait 2 minutes
    - Modify job
  
  - Graphical views (run by the 40% of the users)
    - Graphical impact view
    - Wait 4 minutes
    - Graphical job stream view

The following figure illustrates the concurrency schedule inside Rational Performance Tester V8.2.

Figure 5 - Tivoli Dynamic Workload Scheduler V8.6 Capacity Plan concurrency schedule - details
5 Results

This section shows the results obtained after the completion of each test described in this document. The information is split into distinct subsections, one for each performed test:

- **Capacity Plan** - 100 users running the concurrency schedule, against the 4 Tivoli Dynamic Workload Console V8.6 server configurations; the analysis will focus on nmon collected data for resource usage of Tivoli Dynamic Workload Console V8.6 server, and response times for the significant steps of each scenario.

- **Breaking Point** - for each of the 4 Tivoli Dynamic Workload Console V8.6 server configurations, an increasing number of users has performed the concurrency scenario to find the maximum number of users that the server is able to correctly manage, from a functional point of view, for a given configuration; the analysis will focus on the response times for the significant steps of each scenario, to understand also from the performance/scalability point of view, an acceptable maximum number of users, so that the time for a user to be waited to get any response from the server never goes above a tolerable value.

- **Garbage Collection Policy** – for the configurations C1, C2, and C3, the concurrency schedule run with 100 users changing the garbage collection policy from default optthruput, to gencon and subpool policies; the analysis will focus on response times for the significant steps of each scenario.

5.1 Capacity Plan

This section contains the results obtained after the capacity plan test activity.

It can be useful to remember that this test consists of running the concurrency schedule, with the fixed number of 100 concurrent users, against the Tivoli Dynamic Workload Console V8.6 different server configurations C1, C2, C3, and C4.

Best results, in terms of minimum average CPU usage during the test run, can be obtained with the configuration C3, which is the one provided with 8 CPUs (32 logical cores).

Good results can also be obtained with the configuration C2 (the one provided with 4 CPUs – 16 logical cores), with an average CPU usage that is just a little bit higher than configuration C3.

Configuration C4 has an average CPU usage that is better than C2, but worse than C3. This means that, doubling the RAM and the heap size (initial and maximum values) has no advantage.

The configuration requiring less RAM is the C1; memory usage increases as the configuration power increases.

Configurations C1, C2, and C3, the ones provided with 8.0 GB of RAM, have a similar behavior for memory usage, while configuration C4, the one provided with 16.0 GB of RAM, shows a small increase.
To obtain best results, in terms of minimize memory usage, choose a configuration provided with 8.0 GB of RAM, with the configuration with less CPUs (C1) having a smaller improvement than the ones provided with more CPUs (C2 and C3).

The configuration C4 shows a little increase of RAM usage, but it can be considered acceptable because it is limited to about 20 MB.

For the disk read activity, it is possible to assert that it is very limited during the capacity plan test run, with some peaks in the initial phase and a flat behavior in the middle and final phase.

In more detail, configurations C3 and C4 display the most significant activity, with some peaks that are limited to about 250-300 KB/sec.

The best results, in terms of minimum disk read activity, can be obtained with the configuration C2; but any other configuration presents a very reduced disk read activity; if the target is the disk read activity decrease, any configuration can be chosen, or delegate the choice of the suitable configuration to another feature.

For the disk write activity, there is no evidence of a configuration to get best performances, in terms of smallest amount disk write activity, because the average values are very similar.

To avoid peaks, configurations C1 and C4 should be used, but this does not seem to provide strong advantages compared with configurations C2 and C3.

The network read activity has a very similar trend for all the configurations. This is expected because it should depend on the number of scheduling objects the Tivoli Dynamic Workload Console V8.6 exchanges with the Tivoli Workload Scheduler V8.6 server, which remains fixed for all the configurations.

To obtain the best results, in terms of minimum network read usage, any configuration can be chosen, even the weakest one C1; if no peak is desired, looking at the maximum network read values recorded, configurations C1 and C4 should be preferred, but even the remaining ones display similar trends.

Network write activity shows very similar trends for all the configurations. This is expected because it depends on the number of scheduling objects the Tivoli Dynamic Workload Console V8.6 exchanges with the Tivoli Workload Scheduler V8.6 server, which remains fixed for all the configurations.

Average network write activity is very close for all the configurations.

To obtain best performances in terms of minimum network write activity, any configuration can be chosen; if it is also needed to avoid peaks, then configuration C1 should be discarded, and configuration C3 should be chosen because it has the smallest maximum value.

The next part of this section is related to the response times. It can be useful to remember that the analysis of the response times is focused on a subset of most significant steps that can be found in all the test scenarios of the concurrency schedule. For each test scenario, one or more significant steps were identified, considering that a step should be meaningful for the related scenario and it can have response times higher than expected.

The following table summarizes the main steps identified for each test scenario.
<table>
<thead>
<tr>
<th>Test scenario</th>
<th>Category</th>
<th>Main step</th>
<th>Other significant steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>All job streams in success</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Log in, monitor job streams, dependencies</td>
</tr>
<tr>
<td>All job streams in wait</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Log in, monitor job streams</td>
</tr>
<tr>
<td>All jobs in success</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Log in</td>
</tr>
<tr>
<td>All jobs in error</td>
<td>Monitoring</td>
<td>Query task</td>
<td>Log in</td>
</tr>
<tr>
<td>Create jobs and job stream</td>
<td>Modeling</td>
<td>Save job stream</td>
<td>Log in, create workload definitions, Select engine, edit job</td>
</tr>
<tr>
<td>Modify job</td>
<td>Modeling</td>
<td>Perform query</td>
<td>List workload definitions</td>
</tr>
<tr>
<td>Graphical Impact view</td>
<td>Graphical view</td>
<td>Graphical Impact view</td>
<td>Login, graphical task</td>
</tr>
<tr>
<td>Graphical Job stream view</td>
<td>Graphical view</td>
<td>Graphical Job stream view</td>
<td>Login, graphical task</td>
</tr>
</tbody>
</table>

Table 11 - Capacity plan - concurrency schedule test scenarios main steps

The total number of monitored steps is 16.

It is interesting to analyze the results coming from the previous analysis:

- For “login” the best response times can be obtained with configurations C2 or C3 (almost identical response times); configuration C1 has the weakest response time, while configuration C4 is slightly worse than C2 and C3.

- The query “all jobs in error” (returning 30 jobs), has the best response time with configuration C3 and C2; configuration C4 has a good response time, but not as good as C3 and C2; configuration C1 has the weakest response time.

- The query “all jobs in success” (returning 10,000 jobs), has the best performance (minimum response time) with configuration C4 and C2; configurations C1 has a slightly worst response time, while configuration C3 has the highest response time. It is important to note that all the response times are limited in a very small time interval. Another important fact is that most of the activity needed to complete this query step is performed by the Tivoli Workload Scheduler V8.6 engine; this is why the response times seem not to be correlated with the Tivoli Dynamic Workload Console V8.6 server configuration power.

- The query “all job streams in wait” (returning 98 job streams), has the best response time with configuration C3. Configuration C2 also provides good performances, while configuration C4 has a higher response time; the weakest performances were obtained with configuration C1.

- The query “all job streams in success” (returning 400 job streams), has the best response times with configurations C3 and C2; configuration C4 has a slightly higher response time; while the weakest performances were obtained with configuration C1.

- The step to retrieve the dependencies of a job stream returned in the result set of “all job streams in success” query task, has a stable response time, as the configuration changes.
The “monitor job streams” step performed either in “all job streams in success” or in “all job streams in wait” test scenarios, has the best performances with configurations C3 and C2; configuration C4 offers high response times, while the weakest response time was obtained with configuration C1.

The following two figures provide details about response time comparison, for all the Tivoli Dynamic Workload Console V8.6 server configurations (C1, C2, C3, and C4), for a query scenario step ("All job streams in wait" – 98 job streams) and for a graphical view scenario step (the "Impact View" step to get the impact graphical view of a given job stream).

![Figure 6 - All job streams in wait response time - comparison for all the configurations](image_url)
It can be observed that best response time can be obtained with configuration C3; configuration C2 offers a good response time too, just a little bit higher than C3.

Configuration C4 has an intermediate response time, while C1 offers the highest response time being too weak for the workload.

From the previous analysis, it is possible to affirm that, when the task depends mostly on the Tivoli Dynamic Workload Console V8.6, the best results were obtained with configuration C3, but very good results were obtained also with configuration C2.

Configuration C4, even if provided with the most powerful hardware (16.0 GB of RAM and the maximum number of CPUs, that is 8), often has performances that are better than C1, but worse than C3 and C2.

Configuration C1 always has the weakest performances.

This means that the most suitable hardware for Tivoli Dynamic Workload Console V8.6, for monitoring (and login) test scenarios, is the configuration C3 (AIX 7.0.0.0, 8 CPUs, 8.0 GB RAM, initial heap 2.0 GB, maximum heap 4.0 GB); configuration C2 has very good performances, close enough to the ones provided by configuration C3.

Configuration C4 is unnecessary, because it is the most expensive, having 16.0 GB of RAM, and does not bring any advantage over using 8.0 GB of RAM. This is because using a heap of 4.0 GB as initial value and 8.0 GB as maximum value, can produce objects too bulky to be managed, affecting the Tivoli Dynamic Workload Console V8.6 performances.
Configuration C1 is not enough to manage the provided workload and user concurrency; then, 2 physical CPUs are not enough for this kind of test activity.

From the previous analysis, it is also possible to assert that:

- The “save job stream” step, performed during the “create jobs and job stream” test scenario, requires a very short time, which does not depend on the particular configuration of the Tivoli Dynamic Workload Console V8.6 server. Any configuration provides the best performances in terms of minimum response time.

- The “create workload definition” step has the best performance with configuration C3; configuration C2 also provides very good results, while configuration C4 has the weakest performances. Configuration C1 has very high response times (4 times greater than best response time obtained with C3 configuration).

- The “select engine” step has the best response time with configuration C3; while configuration C2 provides very similar response time; configuration C4 has a response time that is about twice the C2 response time; the worst results were obtained with configuration C1, about 3 times higher than configuration C2.

- The “edit job” and “list workload definitions” steps have the best results with configuration C3; configurations C1 and C2 offer good results anyway, while configuration C4 has the weakest performance.

- The “perform query” step has the best results with configuration C3; configurations C1 and C2 offers good results anyway, while configuration C4 has the weakest performance.

Similar considerations can be done also for the modeling steps; best performances were obtained with configuration C3; configuration C2 offers also very good performances; configuration C1 and C4 have the weakest response times.

To get the best performances for modeling steps, use configuration C3 or, as a cheaper alternative (paying a small worsening in response times), C2. Configurations C1 and C4 offer worst performances and should be discarded here.

It is possible to note that:

- The “graphical impact view” step has the best response time with configuration C2 and C3; configuration C4 offers a slightly worse response time, while the weakest response time was obtained with configuration C1.

- The “graphical job stream view” step has the best response time both with configurations C2 and C3; configuration C4 has a response time of inferior quality; the worst response time was obtained with configuration C1.

- The “graphical task” step, run in both graphical test scenarios, has the best response time with configuration C3; configuration C2 offers good performances too; configuration C4 offers performance of weaker quality; the weakest performance was obtained with configuration C1 (about 4 times worse than best performance with configuration C1).

For the graphical steps, the best results can be obtained with configuration C3; configuration C2 also offers good performances (even if not as good as C3); although configuration C4 is provided with the most powerful hardware, its performances are always weaker than C2 and C3; configuration C1 is not enough to manage 100 concurrent users with the provided workload in terms of scheduling objects.
5.1.1 Conclusions and recommendations

The conclusions that can be obtained from the Tivoli Dynamic Workload Console V8.6 Capacity Planning test activities are the following: all the tests were completely and successfully performed from a functional point of view. No exceptions occurred and all the verification points were successfully matched.

From a performance point of view, with the only exception of the “all jobs in success” query task that has a very big result set, and for which the most of the response time is used by the Tivoli Workload Scheduler V8.6 engine, all the analyzed test scenarios have good response times, when the Tivoli Dynamic Workload Console V8.6 is running on the configuration C2 or C3.

Configuration C1 is not enough to manage 100 concurrent users with the provided workload: even if from a functional point of view there were no issues found, the performances were not brilliant, 4 times worse than the corresponding results obtained with configuration C2 and C3. Two physical CPUs are not enough to successfully complete, from a performance point of view, the Tivoli Dynamic Workload Console V8.6 Capacity Plan.

Configuration C4 demonstrates that freely increasing the heap size (both initial and maximum), does not produce improvements to the performance but it affects them by getting response times that are weaker than the lower configurations (C2 and C3).

The recommendation, to obtain best performances for the Capacity Plan activity, for a user needing to build up a Tivoli Dynamic Workload Console V8.6 environment that needs to manage a similar workload (in terms of scheduling objects and/or concurrent users), is to use configuration C3. If, for any reason this configuration cannot be used, even using configuration C2 will produce very good results. Configuration C4 is to discard because it is the most expensive and produces results worse than the cheaper configurations C2 and C3. Finally, configuration C1 is to be discarded because, even if the functional correctness is granted, it provides weak performances, compared to C2 and C3.

The following table summarizes the obtained results.

<table>
<thead>
<tr>
<th>Number of users</th>
<th>Test</th>
<th>Best configuration for performance</th>
<th>Best configuration for performance/cost</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Capacity Plan</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Good results with best performance configuration; Acceptable results with best performance/cost configuration.</td>
</tr>
</tbody>
</table>

Table 12 - Capacity Plan results

5.2 Breaking point

This section presents the breaking point test activities results. The goal of this test is to identify, for a given configuration the “breaking point” under the concurrency schedule run that is the maximum number of concurrent users the Tivoli Dynamic Workload Console V8.6 is able to manage without errors.
In this analysis, the breaking point is considered mostly from a functional point of view. The upper bound is defined as the maximum number of concurrent users that are able to complete the test scenarios run successfully (no errors, exceptions or irregular behaviors).

Tests were run, on any configuration (C1, C2, C3, and C4) by running the concurrency schedule with a growing number of users starting from 100 users and rising up to 300 users.

Result analysis focuses on the previously identified 16 most significant steps coming from the test scenarios that compose the concurrency schedule.

5.2.1 Conclusions and recommendations

The following figures provide details about the response time vs. users trend, for 2 distinct scenarios (“All job streams in success”, “Graphical job stream view”).

Figure 8 - All job streams in success response time for different concurrent users, all configurations comparison
The previous analysis demonstrates that configurations C2 and C3 allow good scalability up to 200 users, and tolerable scalability up to 250 users.

From 250 users up to 300 users, these configurations grant a functional accuracy (every task is successfully completed by all the concurrent users), but the response times are weak and often not tolerable.

Configuration C4, even being the most powerful one, offers performances that are always weaker than configurations C3 and C2; this means that further increasing the heap size of the Tivoli Dynamic Workload Console V8.6 server does not provide any advantage but affects the performance.

Configuration C1 provides the weakest performances nearly always; this means that this configuration is not powerful enough for the given workload in terms of scheduling objects and concurrent users.

Then, the conclusion is that it is possible to scale up to 200 users with good performances, for a user willing to build up a scheduling environment similar to the one used in this test activity, and up to 250 users with tolerable performances, using a configuration very close to C2 or C3.

Then, a system AIX 7.0.0.0, with 4 or 8 CPU cores, and 8.0 GB of RAM (with initial heap size of 2.0 GB, and maximum heap size of 4.0 GB), allows a good scalability for the given workload.

In more detail, focusing on the optimal configurations C2 and C3, it is possible to state that:

- Login has good response times for up to 200/250 users
• All jobs in error has good response times up to 200 users, tolerable at 250 users
• All jobs in success has high response times but this is due to the very high amount of objects in the result set (10.000); scalability is linear up to 250 users
• All job streams in wait has tolerable response times up to 250 users
• All job streams in success has very good response times up to 250 users
• Dependencies has good response times up to 250 users
• Monitor job streams has tolerable response times up to 250 users
• Graphical Impact view has good response times up to 250 users
• Graphical Job stream view has almost good response times up to 250 users
• Graphical task has good response times up to 250 users
• Save job streams has excellent response times up to 250 users, and very good response time up to 300 users
• Create workload definitions has tolerable response times up to 250 users
• Select engine has tolerable response times up to 250 users
• Edit job has good response times up to 250 users
• List workload definitions has quiet good response times up to 250 users
• Execute query has good response times up to 250 users

In details (focus is on configurations C2 and C3), the login response times are good up to 250 users.

The analyzed monitoring steps sometimes have not very good response times, but this is due to how the Tivoli Workload Scheduler V8.6 engine manages the access to the symphony file required to get the desired scheduling objects.

The graphical views have good response times and linear trends.

The modeling steps have excellent response times (database operations).

The following table summarizes the conclusions obtained after the test analysis:

<table>
<thead>
<tr>
<th>Number of users</th>
<th>Test</th>
<th>Best configuration for performance</th>
<th>Best configuration for performance/cost</th>
<th>Notes</th>
</tr>
</thead>
</table>

27
### Table 11 - Breaking point results

<table>
<thead>
<tr>
<th></th>
<th>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</th>
<th>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</th>
<th>Good results with best performance configuration. Acceptable results with best performance/cost configuration. Acceptable results for both the configurations. Acceptable results with best performance configuration. Tolerable results with best performance/cost configuration. Only functionality is granted for both the configurations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>100</strong> Scalability/Breaking point</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>200</strong> Scalability/Breaking point</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>250</strong> Scalability/Breaking point</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>300</strong> Scalability/Breaking point</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 5.3 Garbage collector policies comparison

The garbage collector policies comparison test activity was run comparing the response times only for the 16 previously-identified most significant steps of the test scenarios of the login inside the 3 identified categories of monitoring, modeling, and graphical view.

In this chapter results will be detailed for the configurations C2 and C3 only, because they have the best performances for the concurrency schedule.

It can be useful to remember that this activity was performed with a fixed number of 100 concurrent users running the concurrency schedule, with the garbage collector policy set to the optthruput (default) policy, then to the gencon policy, and lastly to the subpool policy.

For monitoring scenarios, no policy seems to have an overall better response time. It is possible to let the default optthruput policy, because changing the garbage collector policy does not give any advantage.

For modeling scenarios, it is possible to note that the default optthruput policy seems to have, in most cases, better response times than the other policies being compared.
Also for the login and graphical view steps, no garbage collector policy seems to have a definitely better response time for all the steps, so that it is possible to keep the default optthruput policy, because changing the garbage collector policy does not give any advantage.

5.3.1 Conclusions and recommendations

The garbage collection comparison test activity, for the Java process of the embedded WebSphere Application Server V7.0.0.0 (FP15) of the Tivoli Dynamic Workload Console V8.6, has highlighted that no policy seems to provide particular advantages to the response times of the 16 analyzed steps, compared with the remaining ones.

In the case of the modeling steps, a small trend can be seen, with a lot of steps where the default optthruput policy seems to have small improvements compared with gencon and subpool policy.

6 Conclusions

All the Tivoli Dynamic Workload Console V8.6 Capacity Plan test activities were successfully run and completed, either from a functional or performance/scalability point of view.

The capacity plan test activity has highlighted that configurations C2 and C3 are the most suitable to manage the input workload, in terms of concurrent schedule, users and test scenarios.

The breaking point test activity has demonstrated that, configuration C2 or C3, can scale up to 200/250 users with all the test scenarios successfully completed.

The garbage collection comparison has shown that changing the policy to subpool or gencon does not provide any advantage to the performance of the Tivoli Dynamic Workload Console V8.6, and it is a good practice to keep the default optthruput garbage collector policy.

Another interesting information that was discovered during the whole test activity is that configuration C1 is not powerful enough to manage the provided input workload, and that increasing the heap size (both the initial and the maximum size), as the physical RAM available increases, does not provide advantages but, on the contrary, it can affect the Tivoli Dynamic Workload Console V8.6 performances such as response times and scalability.

The following table summarizes the most interesting thing found during the entire test activities performed.

<table>
<thead>
<tr>
<th>Number of users</th>
<th>Test</th>
<th>Best configuration for performance</th>
<th>Best configuration for performance/cost</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Capacity Plan</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Good results with best performance configuration; Acceptable results with best performance/cost configuration.</td>
</tr>
</tbody>
</table>
## Table 13 - Test activities results

<table>
<thead>
<tr>
<th>100</th>
<th>Garbage Collector tuning</th>
<th>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</th>
<th>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</th>
<th>Use the default opttrhuput policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Scalability/Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Good results with best performance configuration; Acceptable results with best performance/cost configuration.</td>
</tr>
<tr>
<td>200</td>
<td>Scalability/Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Acceptable results for both the configurations;</td>
</tr>
<tr>
<td>250</td>
<td>Scalability/Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Acceptable results with best performance configuration; Tolerable results with best performance/cost configuration.</td>
</tr>
<tr>
<td>300</td>
<td>Scalability/Breaking point</td>
<td>C3 (AIX 7.1.0.0, 8x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>C2 (AIX 7.1.0.0, 4x PowerPC_POWER7 3300 MHz, RAM 8.0 GB, Swap 2.0 GB, HDD 120 GB)</td>
<td>Only functionality is granted for both the configurations</td>
</tr>
</tbody>
</table>
**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [http://www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.