Workload Scheduler Version 9.4 Fix Pack 3
Performance Report

an IBM + HCL product

Document version 1.0

Pier Fortunato Bottan
Giorgio Corsetti
Workload Automation Performance Team - HCL Rome Lab
Contents

List of Figures ................................................................. 3
List of Tables ....................................................................... 3

1.  Introduction ....................................................................... 5
1.1.  What’s new since version 9.4 ........................................... 5

2.  Scope .................................................................................. 5
2.1.  Executive summary .......................................................... 5

3.  Performance Test ............................................................... 6
3.1.  Test Approach ................................................................. 6
3.2.  Environment ................................................................. 6
3.3.  Test tools ................................................................. 12
3.4.  Test Benchmarks and Results ........................................... 12
  3.4.1.  Scheduling Workload .................................................. 12
  3.4.2.  Plan View ............................................................... 19
  3.4.3.  Job Stream View ....................................................... 20
  3.4.4.  High network latency test ........................................... 21

4.  Best Practices ...................................................................... 22
4.1.  Scheduling ...................................................................... 22
  4.1.1.  Scheduling using event rules: Event Processor Throughput .... 22
  4.1.2.  Scheduling using file dependencies ................................ 24
  4.1.3.  Scheduling using a start condition ............................... 25
  4.1.4.  Scheduling using conman sbs ..................................... 25
4.2.  Dynamic domain manager table cleanup policy ................... 25

5.  Recommendations ............................................................. 27
5.1.  CPU capacity ............................................................... 27
5.2.  Storage ........................................................................... 28
5.3.  Memory ........................................................................... 28
5.4.  Tunings and settings ....................................................... 29
  5.4.1.  Data Source ............................................................. 29
  5.4.2.  Plan replication in the database (mirroring) .................... 29
  5.4.3.  Oracle database configuration ..................................... 30
  5.4.4.  Comprehensive configuration and tuning ....................... 30

6.  Capacity Plan Examples ...................................................... 32

7.  Notices ............................................................................... 34
7.1.  Trademarks ..................................................................... 35

List of Figures

Figure 1. Overall deploy view of test environment ............................... 7
Figure 2. Dynamic Workload Console node configuration ...................... 8
Figure 3. Master Domain Manager node configurations ................................................................. 9
Figure 4. Database node configurations .................................................................................................. 10
Figure 5. Storage Solution .......................................................................................................................... 11
Figure 6. Dynamic agent daily throughput. The total number of jobs scheduled in a day ......................... 14
Figure 7. Zoomed view of dynamic agent scheduling .............................................................................. 15
Figure 8. Dynamic agent schedule: ad hoc submission with 1000 scheduled jobs ................................. 15
Figure 9. Dynamic agent schedule: jobs with internal and external conditional dependencies ............... 16
Figure 10. Dynamic agent schedule: jobs belonging to a critical path .................................................. 16
Figure 11. Fault tolerant agent schedule throughput in daily workload (10^5 jobs) .................................... 17
Figure 12. Average job schedule delay over time ....................................................................................... 17
Figure 13. Job plan status update delay over time ....................................................................................... 18
Figure 14. Total average CPU utilization at master domain manager and database .......................... 18
Figure 15. Job Stream network for the plan view scenario ..................................................................... 19
Figure 16. Comparisons of rendering time for graphical view reload ..................................................... 20
Figure 17. Network latency impact on Dynamic Domain Manager throughput capabilities .................. 21
Figure 18. Network latency impact on jobs plan status update ................................................................. 22
Figure 19. Number of actions triggered by the event processor ............................................................... 23
Figure 20. Dynamic Domain Manager job submission throughput in the file creation ......................... 24
Figure 21. MDM CPU utilization comparison with and without event rule processing ........................ 24
Figure 22. File dependency releases in a 1200 jobs/min workload as baseline ........................................ 25
Figure 23. Plan update delay while dynamic job table is being cleaned up .............................................. 26
Figure 24. Database Server disk busy while dynamic job table is being cleaned up ............................... 26
Figure 25. Impact Dynamic Domain Manager throughput capabilities .................................................. 27
Figure 26. Example of Job Stream that handles the cleanup of dynamic jobs ......................................... 27
Figure 27. I/Ozone benchmark output ..................................................................................................... 28

**LIST OF TABLES**

Table 1. Software level of code .............................................................................................................. 6
Table 2. Test tools ................................................................................................................................. 12
Table 3. Daily plan workload composition ............................................................................................... 13
Table 4. Job Stream filtered value for the Plan View scenario ............................................................... 19
Table 5. Event processor capacity in terms of maximum throughput (events per minute) ....................... 23
Table 6. Dynamic Workload Console WebSphere Application Server heap configuration .................. 28
Table 7. Engine WebSphere Application Server heap configuration ...................................................... 29
Table 8. Main configurations and tunings ............................................................................................... 32
Table 9. Capacity planning samples ....................................................................................................... 33
1. Introduction

Workload Scheduler is a state-of-the-art production workload manager, designed to help customers meet their present and future data processing challenges. It enables systematic enterprise-wide workload processing for both calendar and event-based (real-time) workloads across applications and platforms. Workload Scheduler simplifies systems management across distributed environments by integrating systems management functions. Workload Scheduler plans, automates, and controls the processing of your enterprise’s entire production workload.

Pressures in today’s data processing environment make it increasingly difficult to guarantee a high level of service to customers. Many installations find that their batch window is shrinking. More critical jobs must be finished before the workload for the following morning begins. Conversely, requirements for the integrated availability of online services during the traditional batch window put pressure on the resources available for processing the production workload.

Workload Scheduler simplifies systems management across heterogeneous environments by integrating systems management functions.

1.1. What’s new since version 9.4

In the last year and half, three different fix packs have been released for the 9.4 major release.

For more details about new features introduced with these fix packs, see the Summary of enhancements in the online product documentation in IBM® Knowledge Center:

- 9.4 FP1
- 9.4 FP2
- 9.4 FP3

2. Scope

2.1. Executive summary

The objective of the tests described in this document is to report the performance results for the new version of the product, V9.4.0.3, executed in a test environment based on the VMWare ESX® - Linux x86 platform with a comparable resource assignment to the previous performance environment (see Workload Scheduler v9.4 performance report) which was based on the POWER7® - AIX® platform.

The performance results could be summarized as follows:

- Consolidate previous performance achievements in terms of throughput.
- New tunings for software performance improvements considering the new test environment architecture.
- New key performance indicators.
- New improvements for job stream and plan view graphical interfaces.
3. Performance Test

3.1. Test Approach

As specified in section 2.1, most of performance test focus was specific for the new performance test environment used to validate the Workload Scheduler 9.4.0.3 release. The guideline was to keep the performance benchmark results, collected in previous releases and on different platforms, as key performance indicators. Scheduling throughput, resource consumption and reliability are continuously certified assuring no degradation with respect previous releases. Specific tests were implemented for validating performance improvements like the “Plan View” and “Job Stream View” graphical views.

In addition, other specific workloads were benchmarked in the performance environment in accordance with input received during continuous Workload Scheduler customer interactions. The latter is continuously tested with a daily plan running every day on two production-like environments (one with IBM DB2® and the second with Oracle database®).

In this context, continuous monitoring was applied with special focus on key performance indicators (scheduling and mirroring throughput, average delays, internal queues sizes) and on the main hardware resources (CPU, memory, disk busy) to prevent memory leaks, unexpected hardware consumption and product performance degradation during long run workload scenarios.

3.2. Environment

The test environment was based on virtual machines hosted on VMware ESXi servers running on Dell™ PowerEdge R630 Intel™ Xeon(R) CPU E5-2650 v4 @ 2.20GHz. All tests were performed in a 10 GB local area network.

The following table summarizes the software used and the version:

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Linux Red Hat® server 7.3</td>
</tr>
<tr>
<td></td>
<td>Kernel 3.10.0-514</td>
</tr>
<tr>
<td>RDBMS</td>
<td>IBM DB2 v11.1.2.2</td>
</tr>
<tr>
<td></td>
<td>Oracle 12c Enterprise Edition 12.1.0.1.0</td>
</tr>
<tr>
<td>J2EE</td>
<td>IBM WebSphere® Application Server 8.5.5.13</td>
</tr>
<tr>
<td></td>
<td>with IBM Java 8.0.5.6</td>
</tr>
<tr>
<td>LDAP</td>
<td>IBM Directory Server 7.2</td>
</tr>
<tr>
<td>Jazz™ for Service Management</td>
<td>JazzSM 1.1.3 DASH 3.1.3 CP6</td>
</tr>
<tr>
<td>WA</td>
<td>9.4.0.3</td>
</tr>
</tbody>
</table>

Table 1. Software level of code

The HTTPS protocol was used and an IBM HTTP Server with IHS WebSphere Application Server Plugin acted as a load balancer with “Random” policy to distribute user load on the Dynamic Workload Console servers. The procedure described in the following link was followed to set up a high availability configuration (also known as “cluster” elsewhere in this document):

Figure 1. Overall deploy view of test environment
Figure 2. Dynamic Workload Console node configuration.
Figure 3. Master Domain Manager node configurations
Figure 4. Database node configurations
Figure 5. Storage Solution
3.3. Test tools

The main tools used in this performance test context are listed in the following table.

<table>
<thead>
<tr>
<th>tool</th>
<th>scope</th>
<th>version</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOzone</td>
<td>Benchmark I/O capacity</td>
<td>3.434</td>
<td><a href="http://www.iozone.org">www.iozone.org</a></td>
</tr>
<tr>
<td>Perfanalyst</td>
<td>DB2 and WAS tuning</td>
<td>1.1.4</td>
<td><a href="https://www.ibm.com/developerworks/community/groups/community/perfanalyst">https://www.ibm.com/developerworks/community/groups/community/perfanalyst</a></td>
</tr>
<tr>
<td>ROOT framework</td>
<td>Data analysis and Presentation</td>
<td>6.0.8</td>
<td><a href="https://root.cern.ch/">https://root.cern.ch/</a></td>
</tr>
<tr>
<td>IBM Support Assistant</td>
<td>Java Core, Garbage Collector and thread dump analysis</td>
<td>5.0.2.4</td>
<td><a href="https://www-01.ibm.com/software/support/isa/">https://www-01.ibm.com/software/support/isa/</a></td>
</tr>
<tr>
<td>netem</td>
<td>Network delays emulator of wide area network</td>
<td>Kernel 3.10.0-514</td>
<td><a href="https://wiki.linuxfoundation.org/networking/netem">https://wiki.linuxfoundation.org/networking/netem</a></td>
</tr>
</tbody>
</table>

Table 2. Test tools.

3.4. Test Benchmarks and Results

3.4.1. Scheduling Workload

This section reports the details of the workload included in the daily production plan deployed in the Performance Test environments. The workload is distributed among fault tolerant and dynamic agents. The total number of jobs that are executed daily is around 530000 jobs.
Table 3. Daily plan workload composition

This workload is used as a standard benchmark for establishing key performance indicators whose baseline is continuously verified to track performance enhancements.
Figure 6. Dynamic agent daily throughput. The total number of jobs scheduled in a day is around $4.3 \times 10^5$

The workload represented in Figure 6 is the daily load of dynamic agent scheduling. In particular the jobs from 18:00 to 23:00 are standard schedules with “every” option. The workload from 14:30 to 17:10 has different components as can be observed from the zoomed view of Figure 7.
Dynamic Agent Throughput

Figure 7. Zoomed view of dynamic agent scheduling

Figure 8. Dynamic agent schedule: ad hoc submission with 1000 scheduled jobs
Figure 9. Dynamic agent schedule: jobs with internal and external conditional dependencies (3200 total jobs)

Figure 10. Dynamic agent schedule: jobs belonging to a critical path (Workload Service Assurance)
The scheduling throughput represented in this section did not suffer any queuing phenomenon (incoming and outgoing throughput are equivalent).

In fact, the throughput analysis reconfirms the performance and scalability levels assured in previous releases. From the workload scheduler user perspective that means, for instance, no substantial delay in the scheduling of a job when the same job is ready to start (see Figure 12) and no substantial latency in the job and job stream status update from the Dynamic Workload Console (see Figure 13).
Figure 13. Job plan status update delay over time

The above results evidence the promptness of Workload Scheduler in the case of an intensive workload (2600 jobs/min both for dynamic and fault tolerant agents) with a dynamic scheduling delay of less than 1 minute. It is interesting to note how the average scheduling delay is around 4 seconds as expected from the batchman process configuration (bm look, bm read settings). It must be remarked, once again, that these results must be correlated with the test environment and the workload discussed in this context; nevertheless, they could be considered as references while planning a Workload Scheduler deployment.

Figure 14 shows the CPU resource utilization trend with respect to the outgoing throughput for dynamic agent scheduling, including two scenarios, one exclusively with dynamic agent scheduling and another with both dynamic agents and fault tolerant agents (in the latter case, the throughput must be considered double). As already revealed in previous performance reports, most of the computation on the master domain server node is related to dynamic job scheduling. The rest of the computation is related to plan updates and, in this case, also the weight on the resource utilization of fault tolerant agent job result notifications.

Figure 14. Total average CPU utilization at master domain manager and database nodes sides vs different workload
3.4.2. Plan View

The graphical representation of the plan, the Plan View, was redesigned to enhance the user experience (UX). The new design helps the user accomplish tasks easily and efficiently. Simple shapes to easily identify objects have been created. New icons to improve the interaction were designed, and help quickly identify actions you can perform. New colors and a background to better visualize the objects have been applied.

The new graphical views (including the Plan View) have been implemented within a new client base framework. Most of the previous master workload has been displaced to the client browser. That includes object relationship computation and graphical rendering. This important architectural change increases concurrency for Workload Scheduler operators accessing new graphical views.

Several job stream types were tested as reported in the following table:

<table>
<thead>
<tr>
<th>Workload</th>
<th>Number of Job Streams</th>
<th>Number of Job Stream Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS1</td>
<td>954</td>
<td>2895</td>
</tr>
</tbody>
</table>

Table 4. Job Stream filtered value for the Plan View scenario

![Figure 15. Job Stream network for the plan view scenario.](image)

*Figure 15* gives an idea of the level of complexity in the layout of the job streams used in this benchmark. It could be argued that the feasibility of handling such objects in a graphical framework, anyway, tests where intended to stress the capability.
Given several jobs included in a job stream, the rendering time appears to be almost linearly dependent on the total number of dependencies. This is due to the layout computational time.

Tests were performed on a physical Intel Core i7-4710MQ CPU @ 2.50 GHz ---- 8 logical cores host with the following browser levels:

- Mozilla Firefox® ESR 45.6
- Google Chrome™ 55.0

Results show how performances improve, and how they are less related to the specific browser (Chrome had better performance for this scenario in the previous versions of the product).

Note that the memory consumption for the browser process is also not negligible (around 700 MB).

### 3.4.3. Job Stream View

In the Workload Scheduler 9.4.0.3 release, the Job Stream View in the Dynamic Workload console was optimized to avoid database resource consumption and high page response times in case of user concurrency. To validate the expected performance improvements, a test with 60 concurrent users (using Rational Performance Tester) was performed.

All 60 concurrent users performed the same scenario (a monitoring job stream query to search for a specific job stream to retrieve the job log and to display the Job Stream graphical view) for 2-hours long, acting on different job streams:

- 30 users opened the Job Stream View for a job stream with 1000 jobs and 1000 dependencies and kept the Job Stream View open for 5 minutes, with live update refresh rate set to 5 seconds. In this case, the job stream live update call did not return any status change.
- 30 users opened the Job Stream View for a job stream with 200 jobs and 235 dependencies and kept the Job Stream View open for 5 minutes, with live update refresh rate set to 5 seconds. In this case, the job stream live update call returned some status changes.

The improvements with respect to the previous fix packs are meaningful:

- **Dynamic Workload Console average page response time comparison**
  The response times for all of the pages exercised during the scenario were significantly improved. Specifically, the average response time of the live update call for the job stream with 200 jobs and 235
dependencies was reduced by 93%.
• **Average CPU usage on the database server machine**
The amounts of CPU used on average on the database server was reduced by 99%.

### 3.4.4. High network latency test

Network latency has significant impact in a workload scheduler environment with a heavy workload. This objective of this section is to give a row quantitative estimation of the performance impact due to network latency. It could be easily understood that adding a delay while accessing the TCP layer means to increase the serving time at each internal queue that is related to the network. This fact could be negligible in the case of a few transactions per unit of time but can have a disruptive effect in a context of a workload like the ones described in this document.

The Linux kernel capability *netem* has been used to simulate a large area network behavior and to add latency between nodes in the topology. Using the command

```
tc qdisc add dev "net interface" root netem delay "value"ms "variance"ms
```

![Figure 17. Network latency impact on Dynamic Domain Manager throughput capabilities](image-url)
Network latency impact is directly proportional to the workload (in this case, the baseline was 1200 jobs/min) and inversely to the system capacity. For this reason, it is mandatory to correctly take it into consideration when creating a workload scheduler deployment plan.

From the 2 graphs above, it is clear that, in our test environment and running the workload of 1200 jobs/min for at least 5 hours, we started to have meaningful impacts of Workload Scheduler performance capabilities starting from a network delay of 0.7 msec.

4. Best Practices

4.1. Scheduling

Workload Scheduler software offers many features to perform at best its own objective: orchestrate scheduling flow activities. The principal way to schedule is to have job streams included in the plan, by associating it to a run cycle, during the plan generation activity. In addition, the schedule of jobs and job streams could occur dynamically while the plan is running, using, for example, event rules, conman, start conditions, and file dependencies. Even if the latter give a higher level of versatility to accomplish different business scenarios, there are some recommendations that must be considered before planning to adopt them to orchestrate the scheduler completely in case of a heavy workload.

4.1.1. Scheduling using event rules: Event Processor Throughput

It is possible to have rules that trigger actions like job and job stream submission. These rules could detect, for example, a job status change or file creation events. In all of these cases, the events are sent to the event processor queue (cache.dat). In the case of a status change, the consumer is the batchman process, while in the case of remote file monitoring, the agent itself communicates with the event processor. In all of these cases, the final submission throughput strictly depends on event processor throughput capability.

The event processor is a single thread process and its processing capacity is proportional to the core speed and the I/O. For this reason, it cannot scale.

The benchmark was based on 6000 file creation rules, defined for 4 dynamic agents with more than $1.2 \times 10^5$ files created in one hour. The file creation rate was increased during the test to saturate the event processor capacity. It is meaningful to compare the event processor capacity in two different architectures (Table 5).

Figure 18. Network latency impact on jobs plan status update
Table 5. Event processor capacity in terms of maximum throughput (events per minute) comparison

<table>
<thead>
<tr>
<th>Environment architecture</th>
<th>Events per minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power7 AIX env</td>
<td>70</td>
</tr>
<tr>
<td>VMWare Linux env</td>
<td>390</td>
</tr>
</tbody>
</table>

The test was executed during a 240 x 2 jobs/min schedule for dynamic agents and fault tolerant agents. Figure 19 shows the number of actions that the event processors was able to trigger and, in this case, they are mapped (one to one) to job submission (see Figure 20). The focus of this section was to demonstrate the event processor throughput stressed with the file creation rule. It was evaluated that the type of rule does not impact the throughput. There is a specific scenario, status change event rules, whose throughput is constrained by the total number of event rules (see IBM Workload Scheduler 9.3.0.1 Capacity Planning Guide) causing enqueuing on the monbox.

Figure 19. Number of actions triggered by the event processor
Figure 20. Dynamic Domain Manger job submission throughput in the file creation event rule scenario

While planning this kind of solution, not only the event processor capacity must be considered but also its additional resource utilization in terms of CPU.

![Figure 21. MDM CPU utilization comparison with and without event rule processing](image)

It could be noted that to obtain the same throughput there is an additional CPU consumption of about 30% on the master domain machine.

4.1.2. Scheduling using file dependencies

Workload Scheduler allows the release of dependencies to perform scheduling. These releases could depend on several objects (jobs, job streams, resources), file dependency is often a useful feature to implement many business workflows that must be triggered by a file creation. The workload described in section 3.4.1 already includes a component of this type acted against fault tolerant agents. This feature has a different impact on the performance if used with a dynamic agent. In case of a dynamic agent, the entire mechanism is driven by the dynamic domain manager that is in charge of the continuous check on the file existence status. The polling period is driven by the localopts property present on the Dynamic Domain Manager:

```
bm check file = 300 (300 seconds is the default).
```

It defines the frequency with which the dynamic agent is contacted by server about file status. The server workload throughput is ruled by three parameters:

1. Polling period.
2. Number of file dependencies.
3. Network connection between agents and server.

In the test environment, (with around network latency 0.1 ms), the file check throughput was evaluated to be around 44 seconds to check 1000 files.

It is suggested to keep the ratio (number of file dependencies)/(bm check file) less than 0.7.

This scenario was applied to 2000 jobs scheduled with 2000 different file dependencies. File dependency release was triggered in three blocks of 400, 800 and 800 files when the system was charged with 1200 jobs/min job submission (see Figure 22).
4.1.3. Scheduling using a start condition

It is also possible to schedule jobs using file detection as a trigger handled a by job mechanism. This is known as the Start Condition feature: a job called continues to run until a file match is detected.

The capacity of this job stream submission is strictly related to the global schedule capacity. The advantages include leveraging of job control and monitoring.

4.1.4. Scheduling using conman sbs

The "conman sbs" (or equivalent RESTful calls) command adds a job stream to the plan on the fly. If the network of the added job stream is significantly complex, both in terms of dependencies and cardinality, it could cause a general delay in the plan update mechanism. In this scenario, due to scheduling coherence, all the initial updates pass through the main thread queue (mirrobox.msg) missing the benefit of multithreading. It is extremely difficult to identify the complexity of the network that would cause this kind of queueing, in any case, the order of magnitude is of several hundreds of jobs in the job streams and internal and/or external dependencies.

4.2. Dynamic domain manager table cleanup policy

While the workload increases in terms of the number of dynamic jobs executed per day, the dynamic domain manager historical tables increase accordingly. Data persistency in this table allows to perform job log
retrieval for archived plans. The following parameters, in the JobDispatcherConfig.properties file, define the cleanup policy:

- SuccessfulJobsMaxAge
- UnsuccessfulJobsMaxAge
- MoveHistoryDataFrequencyInMins

By default, the cleanup thread starts after the time specified by “MoveHistoryDataFrequencyInMins” lapses since the last occurrence completion or application server boot and removes jobs from the table accordingly with their status and age. If the job table is large (magnitude $10^6$ rows), and the number of records to delete high (magnitude $10^5$), this activity impacts Workload Schedule performance and throughput capabilities, as shown by the example below.

During 1200 jobs/min constant workload, the dynamic job cleaning starts deleting almost $7\times10^5$ rows of jobs that are 10 days old and in successful state. The delete operation in this case took almost 7 minutes to complete, causing an intensive I/O activity on the database, which impacted overall product throughput capabilities.

![Figure 23. Plan update delay while dynamic job table is being cleaned up](image)

![Figure 24. Database Server disk busy while dynamic job table is being cleaned up](image)
To avoid the behavior described above, it could be suggested to handle the policy in a more controlled way. For instance, a specific job could be used to run the cleanup invoking the built-in Workload Scheduler script:

```
<installation path>/TWA/TDWB/bin/movehistorydata.sh -successfulJobsMaxAge 240
```

The script could be executed on a job scheduled during the appropriate time window, when the daily jobs execution is low. In this case, the following configuration should be applied in the `JobDispatcherConfig.properties`:

- `SuccessfulJobsMaxAge = 360` (15 days)
- `MoveHistoryDataFrequencyInMins = 720` (12 hours)

```
SCHEDULE MDMWS#CLEANUP_DWB_DB
DESCRIPTION "Added by composer."
ON RUNCYCLE RULE1 "FREQ=DAILY;INTERVAL=1"
AT 2345
: EU-HWS-LNX47#CLEANUP_DWB_DB
SCRIPTNAME "/opt/IBM/TWA/TDWB/bin/movehistorydata.sh -dbUsr db2user -dbPwd password
-successfulJobsMaxAge 240 -notSuccessfulJobsMaxAge 720"
STREAMLOGON root
DESCRIPTION "This job is used to invoke the script which performs the cleanup of
old dynamic jobs in the database"
TASKTYPE UNIX
RECOVERY STOP
```

Figure 26. Example of Job Stream that handles the cleanup of Dynamic Domain Manager table entries

### 5. Recommendations

#### 5.1. CPU capacity
All tests described in this document were executed on virtual CPUs assigned exclusively to VMs (reserved resources). The Workload Scheduler product can run successfully with different configuration. However, in the event additional resources are available, or deploying into a virtual environment where over commitment is possible and resources will be dynamically utilized, the recommended values will permit greater concurrency and reduce processor latency. While planning the correct CPU sizing, the information provided in Table 9 could be a reference point to start. The validity of the superposition property that allows us to assume that the resource usage could be considered as the sum of the UI (DWC) usage plus the core scheduling usage was demonstrated.

5.2. Storage

It is not in the scope of this document to suggest a specific storage solution, but the relevance of I/O capacity was underlined in previous performance reports in relation to the overall product performance.

The numbers presented in Figure 27 could be used as a reference while planning a solution, because they are the output of I/O Industry standard benchmark, such as IOzone, and they can be considered key performance indicators to be used for comparison.

![IOzone Benchmark Results](image)

Figure 27. IOzone benchmark output run with “-R -I 5 -u 5 -r 4k -s 100m -F file1 ...file5” options

Moreover, we have monitored VMWare ESXi server disk latency over time while Workload Scheduler daily production plan was running: in average the disk latency was negligible, with some sporadic peaks lower than 2 ms.

5.3. Memory

RAM size is strongly impacted by the JVM heap size settings whose suggested configuration could be found in the following tables:

<table>
<thead>
<tr>
<th>Concurrent users range x DWC node</th>
<th>1 – 50</th>
<th>50 -100</th>
<th>100 - 150</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWC heap size</td>
<td>2 GB</td>
<td>4 GB</td>
<td>6 GB</td>
</tr>
</tbody>
</table>

Table 6. Dynamic Workload Console WebSphere Application Server heap configuration
### 5.4. Tunings and settings

The following parameters were tuned during performance tests. These appliances are based on common performance best practices, also used in previous releases, and tuning activities during the test execution.

#### 5.4.1. Data Source

On the Linux x86 platform, a specific analysis of JDBC data source configuration was performed due to a potential native limitation in socket activity within the J2EE context. To achieve a throughput comparable with AIX, the JDBC data source configuration has been reworked. In the case of a high-volume schedule (for example, greater than 1200 scheduled jobs/min), the following setting is strongly suggested:

- **Validate existing pooled connections** = false

  This setting causes the Application Server to not perform a dummy query to the database to validate a connection each time it is re-used from the JDBC pool.

  Having this setting equal to true is used to catch the StaleConnectionException that could arise getting a held connection from the pool. StaleConnectionException extends the SQL Exception that the application already catches. Handling this exception could make the application able to perform a recovery action on the pool (in Workload Scheduler, the application server is in charge of this activity).

  To limit the probability of encountering StaleConnectionExceptions when the validation of existing pooled connections is set to false, the following additional tunings are suggested:

  - Data source MIN connections set to 0.
  - Data source Unused Timeout no greater than 1/2 the value configured for the firewall timeout, if a firewall is present.
  - Reap Timer value less than the Unused Timeout.
  - Data source Purge Policy set to entire pool.

  Dynamic Domain Manager job submission throughput benefits from this setting. The drawback effect is related to the possibility of encountering an exception while attempting to connect to the DB, in case of unexpected network issues. In any case, the other suggested settings allow to recover the event by recreating the entire pool.

#### 5.4.2. Plan replication in the database (mirroring)

The plan replication feature, also known as mirroring, has been improved release after release by means of a parallelism (multithreading) and caching. The former is defaulted to 6 threads process with 6 related mirrorbox_.msg queues. In case of high rates (thousands of job status updates per minute) or other environment configuration (network latency between Master Domain Manager and Database), it could be advisable to
enlarge the number of mirroring threads and queues:

- com.ibm.tws.planner.monitor.subProcessors = 10

Furthermore, the usage of a cache improves performances in the way the plan update processing avoids querying database for information already handled:

- com.ibm.tws.planner.monitor.cachesize = 40000

Since Workload Scheduler version 9.4.0.1, a new caching mechanism was added to accomplish the stress of scenarios with thousands of file dependency status updates:

- com.ibm.tws.planner.monitor.filecachesize = 40000

These settings are defined in the <WS_INST_PATH>/WAS/TWSProfile/properties/TWSConfig.properties file.

### 5.4.3. Oracle database configuration

The Oracle database configuration that has been used in this context was the default applied by 12c Enterprise Edition 12.1.0.1.0 installation. It is advisable to enable the Datafile AUTOEXTEND property (ON) considering that the settings and workload described in this section caused a table space occupancy of about 50 GB.

### 5.4.4. Comprehensive configuration and tuning

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Workload Console configuration settings repository (see <a href="https://www.ibm.com/support/knowledgenter/SSGSPN_9.4.0/com.ibm.tivoli.itws.doc_9.4/distr/src_ad/awsaddwcanddb2.htm">https://www.ibm.com/support/knowledgenter/SSGSPN_9.4.0/com.ibm.tivoli.itws.doc_9.4/distr/src_ad/awsaddwcanddb2.htm</a>)</td>
<td>Use database as settings repository</td>
<td>It is strongly recommended to adopt this configuration to allow acceptable UI performance</td>
</tr>
<tr>
<td>WebSphere Application Server WC Thread Pool Size</td>
<td>300</td>
<td>Should be adjusted with number of concurrent users accordingly</td>
</tr>
<tr>
<td>WebSphere Application Server JVM max heap = min heap</td>
<td>Required: 4096 for [50, 100] users per node Suggested: 6144 for [100, 150] users per node</td>
<td></td>
</tr>
<tr>
<td>WebSphere Application Server JVM options</td>
<td>-Djava.awt.headless=true -Xdisableexplicitgc -Xgcpolicy:gencon -Xmn1024m</td>
<td>-Xmn parameter value should be ¼ of total heap size. This parameter should be set to 1536m if heap = 6144</td>
</tr>
<tr>
<td>WebSphere Application Server JDBC max Connections</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>Master Domain Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebSphere Application Server WC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thread Pool Size</td>
<td>300</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>localopts</th>
<th>batchman settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>bm check deadline = 0</td>
<td>bm check file = 120</td>
</tr>
<tr>
<td>bm check status = 300</td>
<td>bm check untils = 300</td>
</tr>
<tr>
<td>bm late every = 0</td>
<td>bm look = 5</td>
</tr>
<tr>
<td>bm read = 3</td>
<td>bm stats = off</td>
</tr>
<tr>
<td>bm verbose = off</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JVM arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Djava.awt.headless=true -Xdisableexplicitgc -Xgcpolicy:gencon -Xmn 1024m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required: 4096 for [100, 200] jobs/min</td>
</tr>
<tr>
<td>Suggested: 6144 for &gt;200 jobs/min</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WebSphere Application Server Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGPRIMARY</td>
</tr>
<tr>
<td>LOGFILSZ</td>
</tr>
<tr>
<td>KEEPFENCED</td>
</tr>
<tr>
<td>MAX_CONNECTIONS</td>
</tr>
<tr>
<td>MAX_COORDAGENTS</td>
</tr>
<tr>
<td>STMT_CONC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB (db2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELF_TUNING_MEM</td>
</tr>
<tr>
<td>APPL_MEMORY, APPLHEAPSZ, DATABASE_MEMORY, DBHEAP, STAT_HEAP_SZ</td>
</tr>
<tr>
<td>AUTO_RUNSTATS</td>
</tr>
<tr>
<td>AUTO_STMT_STATS</td>
</tr>
<tr>
<td>AUTO_REORG</td>
</tr>
</tbody>
</table>

For Oracle it must be changed after installation!
### Table 8. Main configurations and tunings

<table>
<thead>
<tr>
<th>Dynamic Workload Broker</th>
<th>JobDispatcherConfig.properties</th>
<th>ResourceAdvisorConfig.properties</th>
<th>TWSConfig.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE_AGE_TRGT_MCR</td>
<td>NPAGES 120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWS_PLN_BUFFPOOL</td>
<td>NPAGES 182000, PAGESIZE 4096</td>
<td>Queue settings</td>
<td></td>
</tr>
<tr>
<td>TWS_BUFFPOOL_TEMP</td>
<td>NPAGES 500, PAGESIZE 16384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWS_BUFFPOOL</td>
<td>NPAGES 50000, PAGESIZE 8192</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Historical data management: MoveHistoryDataFrequencyInMins = 720
- `Queue.actions.0 = cancel, cancelAllocation, cancelOrphanAllocation`
- `Queue.size.0 = 10`
- `Queue.actions.1 = reallocateAllocation`
- `Queue.size.1 = 10`
- `Queue.actions.2 = updateFailed`
- `Queue.size.2 = 10`
- `Queue.actions.3 = completed`
- `Queue.size.3 = 30`
- `Queue.actions.4 = execute`
- `Queue.size.4 = 30`
- `Queue.actions.5 = submitted`
- `Queue.size.5 = 30`
- `Queue.actions.6 = notification`
- `Queue.size.6 = 30`

- `MaxAllocsPerTimeSlot = 1000`
- `TimeSlotLength = 10`
- `MaxAllocsInCache = 50000`
- `com.ibm.tws.planner.monitor.subProcessors = 10`
- `com.ibm.tws.planner.monitor.filecache.size = 40000`
- `com.ibm.tws.planner.monitor.cachesize = 40000`

### 6. Capacity Plan Examples

In the context of this document, the number of key parameters used to identify the workload was kept to a minimum:

1. Number of concurrent users.
2. Number of jobs to be scheduled.
3. Percentage of dynamic jobs to schedule.
With the above input, it is possible to forecast the resources needed to host the version 9.4.0.3 product. Internal fit functions were used to model the workload and resource usage relationship. A 65% CPU usage was the threshold considered before requesting additional core.

In this section, some examples of capacity planning are reported. Remember that all the requirements are related to Linux X86 VM in a VMWare virtualization with reserved resources; nevertheless, this information could be used as a reference point for different platform architectures.

<table>
<thead>
<tr>
<th>NODE</th>
<th>Number of virtual vCPU cores</th>
<th>RAM Capacity (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K jobs (50% FTA +50% DYN) per day (8 jobs/min), 20 concurrent users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Node</td>
<td>WS Engine, RDBMS, DWC</td>
<td>4</td>
</tr>
<tr>
<td>250K jobs (50% FTA +50% DYN) per day (175 jobs/min), 50 concurrent users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Nodes</td>
<td>WS Engine, DWC</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>RDBMS</td>
<td>4</td>
</tr>
<tr>
<td>500K jobs (50% FTA +50% DYN) per day (350 jobs/min), 100+ concurrent users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Nodes</td>
<td>WS-Engine</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>RDBMS</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>DWC</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 9. Capacity planning samples
7. Notices

This information was developed for products and services offered in the U.S.A.

HCL may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to HCL TECHNOLOGIES LIMITED email: products-info@hcl.com

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: HCL TECHNOLOGIES LIMITED PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this HCL product and use of those Web sites is at your own risk.

HCL may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact HCL TECHNOLOGIES LIMITED email: products-info@hcl.com

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL under terms of the HCL License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot
confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

All statements regarding HCL's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All HCL prices shown are HCL's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

### 7.1. Trademarks

HCL, and other HCL graphics, logos, and service names including "hcltech.com" are trademarks of HCL. Except as specifically permitted herein, these Trademarks may not be used without the prior written permission from HCL. All other trademarks not owned by HCL that appear on this website are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by HCL.

IBM and other IBM graphics, logos, products and services are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. 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.

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

VMware's and all VMWare trademarks and logos are trademarks or registered trademarks in the United States and certain other countries.

Dell, EMC, DellEMC and other trademarks are trademarks of Dell Inc. or its subsidiaries in the United States and certain other countries.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo and JBoss are registered trademarks of Red Hat, Inc. in the U.S. and other countries. Linux is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners.

Mozilla and all Mozilla trademarks and logos are trademarks or registered trademarks in the United States and certain other countries.

Google LLC All rights reserved. Google and the Google Logo are registered trademarks of Google LLC.

NETAPP, the NETAPP logo, and the marks listed at [www.netapp.com/TM](http://www.netapp.com/TM) are trademarks of NetApp, Inc.
hello there! I am an Ideapreneur. I believe that sustainable business outcomes are driven by relationships nurtured through values like trust, transparency and flexibility. I respect the contract, but believe in going beyond through collaboration, applied innovation and new generation partnership models that put your interest above everything else. Right now 119,000 ideapreneurs are in a relationship Beyond the Contract™ with 500 customers in 32 countries. how can I help you?