Workload Scheduler Version 9.5.0.2
Performance Report

an IBM + HCL product

Document version 1.0

Pier Fortunato Bottan
Giorgio Corsetti
Alessandro Tomasi
Workload Automation Performance Team - HCL Rome Lab
## CONTENTS

**Contents** .................................................................................................................. 3

**List of Figures** .............................................................................................................. 3

**List of Tables** ............................................................................................................... 4

1 **Introduction** .............................................................................................................. 6

1.1 What’s new since version 9.5 .................................................................................... 6

2 **Scope** .......................................................................................................................... 6

2.1 Executive summary ..................................................................................................... 7

3 **Performance Test** ..................................................................................................... 7

3.1 Test Approach ............................................................................................................. 7

3.2 Environment ................................................................................................................. 7

3.3 Test tools ..................................................................................................................... 13

3.4 Test Benchmarks and Results ..................................................................................... 13

3.4.1 Scheduling Workload .......................................................................................... 13

3.4.2 Dynamic Workload Console Workload ............................................................... 20

3.4.3 Scheduling Workload for IBM Workload Scheduler on IBM Cloud Private. ........ 27

4 **Best Practices** ......................................................................................................... 31

4.1 Scheduling .................................................................................................................. 31

4.1.1 Scheduling using event rules: Event Processor Throughput ................................ 31

4.1.2 Scheduling using file dependencies .................................................................... 33

4.1.3 Scheduling using a start condition ....................................................................... 34

4.1.4 Scheduling using conman sbs ............................................................................ 34

4.1.5 Scheduling using “every” option ......................................................................... 35

4.2 Dynamic domain manager table cleanup policy ....................................................... 35

5 **Recommendations** .................................................................................................. 37

5.1 CPU capacity ............................................................................................................. 37

5.2 Storage ....................................................................................................................... 37

5.3 Memory ....................................................................................................................... 38

5.4 Tunings and settings ................................................................................................. 38

5.4.1 Data Source ......................................................................................................... 39

5.4.2 Plan replication in the database (mirroring) ......................................................... 39

5.4.3 Oracle database configuration ............................................................................. 40

5.4.4 Comprehensive configuration and tuning ............................................................ 40

6 **Capacity Plan Examples** ......................................................................................... 41

7 **Notices** ...................................................................................................................... 43

7.1 Trademarks ................................................................................................................. 44

---

**List of Figures**

Figure 1. Overall deploy view of test environment ......................................................... 8

Figure 2. Dynamic Workload Console node configuration ............................................. 9
Figure 3. Master Domain Manager node configurations.................................................10
Figure 4. Database node configurations........................................................................11
Figure 5. Storage Solution...............................................................................................12
Figure 6. Dynamic agent daily throughput. The total number of jobs scheduled in a day is around 4.3 x10^5.........................................................................................................................15
Figure 7. Zoomed view of dynamic agent scheduling.....................................................16
Figure 8. Dynamic agent schedule: ad hoc submission with 1000 scheduled jobs............16
Figure 9. Dynamic agent schedule: jobs with internal and external conditional dependencies (3200 total jobs)..................................................................................................................17
Figure 10. Dynamic agent schedule: jobs belonging to a critical path (Workload Service Assurance).................................................................................................................................17
Figure 11. Fault tolerant agent schedule throughput in daily workload (10^45 jobs)..........18
Figure 12. Average job schedule delay over time............................................................18
Figure 13. Job plan status update delay over time...........................................................19
Figure 14. Total average CPU utilization at master domain manager vs different workload 19
Figure 15. Monitor Job Scenario with retrieve of job log...............................................21
Figure 16. Plan View and job stream view......................................................................21
Figure 17. Monitor event rule and triggers...................................................................22
Figure 18. Create jobs and job streams.........................................................................23
Figure 19. Edit existing Job Stream...............................................................................23
Figure 20. Concurrent user income................................................................................24
Figure 21. Pages per second attempted during workload.................................................25
Figure 22. First set of response time trend during increasing workload.........................25
Figure 23. Second set of response time trend during increasing workload ......................26
Figure 24. Third set of response time trend during increasing workload.........................26
Figure 25. CPU utilization during DWC concurrent users test. The workload is distributed over two DWC nodes..........................................................27
Figure 26. Monitor Jobs scenario - average response time comparison between 9.5 FP2 and 9.5 FP1..........................................................................................................................27
Figure 27. Yaml file example for persistent storage........................................................28
Figure 28. Dynamic agent daily throughput on ICP environment....................................29
Figure 29. Average job schedule delay over time on ICP environment............................30
Figure 30. Job plan status update delay over time on ICP environment............................30
Figure 31. Number of actions triggered by the event processor........................................32
Figure 32. Dynamic Domain Manger job submission throughput in the file creation event rule scenario..........................................................................................................................32
Figure 33. MDM CPU utilization comparison with and without event rule processing ......33
Figure 34. File dependency releases in a 1200 jobs/min workload as baseline.................34
Figure 35. Plan update delay while dynamic job table is being cleaned up.......................35
Figure 36. Database Server disk busy while dynamic job table is being cleaned up.........36
Figure 37. Impact Dynamic Domain Manager throughput capabilities while dynamic job table is being cleaned up.................................................................36
Figure 38. Example of Job Stream that handles the cleanup of Dynamic Domain Manager table entries...............................................................................................................................37
Figure 39. IOzone benchmark output run with "-R -l 5 -u 5 -r 4k -s 100m -F file1 ..file5" options.................................................................................................................................38
Table 1. Software level of code ................................................................. 8
Table 2. Test tools .................................................................................. 13
Table 3. Daily plan workload composition ............................................. 14
Table 4. Dynamic Workload Console scenarios with distribution (%) and rates per user ................................................. 20
Table 5. Event processor capacity in terms of maximum throughput (events per minute) comparison ......................................................... 31
Table 6. Dynamic Workload Console WebSphere Application Server heap configuration .......................................................... 38
Table 7. Engine WebSphere Application Server heap configuration .................. 38
Table 8. Main configurations and tunings ..................................................... 41
Table 9. Capacity planning samples ............................................................. 42
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 9.5

In the last year and half, the major release 9.5 and two different fix packs have been released.

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:

- 9.5 GA
- 9.5 FP1
- 9.5 FP2

This new major release 9.5 and related fix packs introduced several changes in the Workload Scheduler infrastructure and functional capabilities. Some of these changes have been considered with special focus during Performance Test phase because there was the need to understand how those changes might affect or not the overall performance and scalability of the product, if compared with the previous releases.

In particular, the main focus areas from Performance Testing perspective have been:

- the replacement of IBM WebSphere® Application Server with IBM WebSphere Application Server Liberty Base for Master Domain Manager, Backup Domain Manager and Dynamic Workload Console
- the introduction of the <DATA_DIR> directory, where are located the product data and data generated by IBM Workload Scheduler, such as logs and configuration information
- Dynamic Workload Console evolution based on new architectural foundation of modern front-end technologies while maintaining previous workload logic and processes. With this refurbishment, Dashboard Application Services Hub (DASH) is replaced by a lean, high-performance in-house solution that is now based on a lightweight, highly composable, fast to start, dynamic application server runtime environment, WebSphere Application Server Liberty
- the new capabilities to organize scheduling objects in a hierarchy of folders. This new capability has been initially added for job and job streams only with 9.5 GA but the full support for all scheduling objects has been extended with 9.5 FP2
- deployment of IBM Workload Scheduler on IBM® Cloud Private, an integrated environment for managing containers that includes the container orchestrator Kubernetes, a private image repository, a management console, and monitoring frameworks

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, V9.5.0.2, executed in a test environment based on the VMWare ESX® - Linux x86 platform with the new middleware dependencies in comparison with previous versions (see Workload Scheduler v9.4 Fix Pack 3 performance report).

The performance results could be summarized as follows:

- Consolidate previous performance achievements in terms of throughput
- Validate tunings for new middleware
- New key performance indicators for product deployment via containers

3 Performance Test

3.1 Test Approach

As specified in section 2.1, the main focus for Workload Scheduler 9.5 performance tests was to validate that the new middleware, supporting the product, allows the same results in terms of throughputs and capacity with respect previous releases; to achieve this the scheduling throughput, resource consumption and reliability are continuously certified.

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></th>
<th>Linux Red Hat® server 7.7</th>
<th>IBM DB2 v11.1.4.4 a</th>
<th>Oracle® 12c Enterprise Edition 12.2.0.1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Kernel 3.10.0-1062.1.1.el7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDBMS</td>
<td>IBM DB2 v11.1.4.4 a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J2EE</td>
<td>WebSphere Application Server Liberty Base 20.0.0.3</td>
<td>Oracle® 12c Enterprise Edition 12.2.0.1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenJDK Runtime Environment (1.8.0_232-b09)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDAP</td>
<td>IBM Directory Server 7.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 1. Software level of code

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>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>5.0.2.8</td>
<td><a href="https://www.ibm.com/support/home/product/C100515X13178X21/other_software/ibm_support_assistant">https://www.ibm.com/support/home/product/C100515X13178X21/other_software/ibm_support_assistant</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 550000 jobs per day.

The scheduling objects (such as jobs, job steams, workstations and others used in the daily workload) have been organized in a hierarchy of folders to validate that this new feature doesn’t determine any unexpected performance impacts on product scheduling capabilities.
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.
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)
Throughput Not-Dynamic Agent Scheduling

Figure 11. Fault tolerant agent schedule throughput in daily workload (10^5 jobs)

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 12. Average job schedule delay over time
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 on the MDM with respect to the outgoing throughput for dynamic agent and fault tolerant agents scheduling as described in section 3.4.1. It is interesting to underline how the MDM CPU footprint for 9.5.x is lower than 9.4.x. This means that the 9.5.x MDM spends less CPU than 9.4.x MDM to run the same workload over time.

Figure 14. Total average CPU utilization at master domain manager vs different workload
3.4.2 Dynamic Workload Console Workload

The Dynamic Workload Console has been tested to probe the performance and scalability levels assured by the new architecture for which Dashboard Application Services Hub (DASH) has been replaced by a lean, high-performance in-house solution that is now based on a lightweight, highly composable, fast to start, dynamic application server runtime environment, WebSphere Application Server Liberty.

In particular, the concurrent user tests have been executed using a topology with 2 DWC in clusters (see Figure 2 for details on single DWC node configuration) configured to use the same DWC database created on Oracle 12c Enterprise Edition 12.2.0.1.0 Server.

The workload has been designed to replicate scenarios and transaction rates applied in previous tests. Table describes the scenarios and its weight in the overall payload including user percentage and single user transaction rate.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Brief description</th>
<th>User %</th>
<th>Rate per user (transaction/min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon01_951jobs</td>
<td>Query on job status (Figure 15)</td>
<td>40</td>
<td>0.5</td>
</tr>
<tr>
<td>Mon03_pv</td>
<td>Open plan view and drill down in job stream details (Figure 16)</td>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>Erule01_monitorER</td>
<td>Select an event rule, view details and monitor the triggering (Figure 17)</td>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>Mod01_CreateJ&amp;JS</td>
<td>Create 2 jobs and add then to a new JS (Figure 18)</td>
<td>20</td>
<td>0.5</td>
</tr>
<tr>
<td>Mod02_EditJS</td>
<td>Modify JS (Figure 19)</td>
<td>20</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4. Dynamic Workload Console scenarios with distribution (%) and rates per user

Figure 15-19 represent the details of each Dynamic Workload Console scenario, from login, transaction loop, driven by specific rate, and logout. Multiple data pool sets have been used to achieve variation in the payload, for example in the Mon_01_jobs scenario the plan queries vary filtering and producing different result set from few to thousands of jobs.
Figure 15. Monitor Job Scenario with retrieve of job log

Figure 16. Plan View and job stream view
Figure 17. Monitor event rule and triggers
Figure 18. Create jobs and job streams

Figure 19. Edit existing Job Stream
As said before, the tests have been performed with two Dynamic Workload Consoles pointing to a database for clustering purpose. The workload balancing has been achieved by using the Rational Performance Tester capability to uniformly distribute workload between the two nodes. Tests have been scheduled with increasing payload in terms of concurrent users (Figure 20) with the specific rates and scenarios distribution (Table 4).

The test with 300 users generates an overall concurrency in the steady state of around:

19 pages/seconds

![User Load Graph](image)

**Figure 20. Concurrent user income**

Results show a linear relationship for incoming throughput (concurrent users) and outgoing throughput (page requests fulfilled per second) as illustrated in Figure 21. Linearity has been also confirmed in resource usage.

The pages response time shown in Figure 22, Figure 23 and Figure 24 below are related to the time spent on the DWC Servers to reply to the Rational Performance Tester calls and it doesn’t include the time spent on the browser to process and show data. Moreover, these results have been obtained in a situation for which no significant scheduling workload was running on the back end (Master Domain Manager, Domain Manager and Dynamic Domain Manager). If significant scheduling workload is running, the average pages response time would be worst considering the extra CPU-intensive work running on both the Master Domain Manager and Database machines.
Figure 21. Pages per second attempted during workload

Figure 22. First set of response time trend during increasing workload
Figure 23. Second set of response time trend during increasing workload

Figure 24. Third set of response time trend during increasing workload
The above results show an equivalent behavior in terms of throughput and resource utilization behavior with respect previous releases based on Dash technologies. Moreover, it confirms how the CPU usage is linear within the increasing number of concurrent users on all the main topological nodes.

It is also interesting to highlight how some performance improvements have been added in the Workload Scheduler 9.5 FP2 if compared with 9.5 GA and 9.5 FP1 versions, especially for the average response time of the pages related to the Mon01_951jobs scenario (Monitor Job Scenario with retrieve of job log).

3.4.3 Scheduling Workload for IBM Workload Scheduler on IBM Cloud Private

Starting from 9.5 version, the deployment of Workload Scheduler is available also for IBM Cloud Private.

IBM Cloud Private provides an integrated environment for managing containers that includes the container orchestrator Kubernetes, a private image repository, a management console, and monitoring frameworks. With
IBM Cloud Private, it is possible to deploy the IBM Workload Scheduler components as Helm charts to quickly configure and run them as Docker container applications in a Kubernetes cluster. Moreover, it is possible to manage the IBM Workload Scheduler components from the IBM Cloud Private dashboard or from the command-line interface. For further details about the IBM Cloud Private solution refer to online documentation.

The main scope of the current section is to document which are the overall scheduling performance of an IBM Workload Scheduler installation on IBM Cloud Private solution in a configuration with persistent storage defined on NFS Server solution.

The IBM Cloud Private environment and topology used for the performance scheduling test is described in the table below:

<table>
<thead>
<tr>
<th>ICP Role</th>
<th>OS</th>
<th>CPU</th>
<th>RAM</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Node</td>
<td>Ubuntu 18.04.1 LTS</td>
<td>8</td>
<td>32 GB</td>
<td>500 GB</td>
</tr>
<tr>
<td>Proxy Node</td>
<td>Ubuntu 18.04.1 LTS</td>
<td>2</td>
<td>16 GB</td>
<td>500 GB</td>
</tr>
<tr>
<td>Worker Node 1</td>
<td>Ubuntu 18.04.1 LTS</td>
<td>6</td>
<td>32 GB</td>
<td>500 GB</td>
</tr>
<tr>
<td>Worker Node 2</td>
<td>Ubuntu 18.04.1 LTS</td>
<td>6</td>
<td>32 GB</td>
<td>500 GB</td>
</tr>
</tbody>
</table>

Table 5. IBM Cloud Private topology used to host IBM Workload Scheduler

Also in this case, all tests described in this section were executed on VMs with virtual CPUs and RAM assigned exclusively (reserved resources) and the database used for both the Master Domain Manager and the Dynamic Workload Console components was DB2 v11.1.4.4a installed on a dedicated VM with 8 vCPUs and 32GB RAM.

As persistent volumes for all the product components (Master Domain Manager, Dynamic Workload Console and Dynamic Agent) the NFS Server solution has been used. To avoid issues during the daily activity, it is highly suggested:

- to configure the persistent volume to use the version 3 of the NFS server
- to set the local lock parameter to all
- to assure to have a limited network latency between the NFS Server and the IBM Cloud Private components (in the test environment used, the average network latency between the Worker Nodes and the NFS Server was less than 0.3 ms The considerations done about the network latency impacts in a workload scheduler environment with a heavy workload available in the section 3.4.4 of Workload Scheduler v9.4 Fix Pack 3 performance report are applicable to 9.5.x version too.

The following is an example of .yaml file that shows how the persistent volumes have been configured in this environment:

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: <release_name>-data
labels:
  <volname>: <volume_name>
spec:
  capacity:
    storage: 15Gi
  accessModes:
  - ReadWriteOnce
  mountOptions:
    - nfsvers=3
    - local_lock=all
  nfs:
    path: <nfs_server_path>
    server: <nfs_server_IPaddress>
```

Figure 27. yaml file example for persistent storage
In this configuration, a daily plan of around 46000 jobs distributed among tens of dynamic agents was executed. Note that fault-tolerate agents are not supported by Workload Scheduler installed on IBM Cloud Private.

The workload represented in **Figure 28. Dynamic agent daily throughput on ICP environment** 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 scheduling throughput represented in this section did not suffer any queuing phenomenon (incoming and outgoing throughput are equivalent).

From the workload scheduler user perspective that means, again, no substantial delay in the scheduling of a job when the same job is ready to start (see **Figure 29. Average job schedule delay over time on ICP environment**) and no substantial latency in the job and job stream status update from the Dynamic Workload Console (see **Figure 30. Job plan status update delay over time on ICP environment**), except for a sporadic unexpected peak.
The above results evidence the promptness of Workload Scheduler installed on ICP in the case of a workload with a peak of around 200 jobs/min with a dynamic scheduling delay of less than 1 minute.

It is interesting to highlight that the dynamic scheduling delay and job plan status update delay start to grow if the incoming workload is increased, especially during the job submission peak. This behavior depends on the fact that the Master Domain Manager DATA_DIR (where the Workload Scheduler queues and Symphony file reside) is located in the persistent volume, that is a network-based file system. In this specific configuration, the network bandwidth and the network latency play a key role in limiting the product throughput capabilities if compared with an on-prem installation for which the Master Domain Manager DATA_DIR is typically defined on a local attached disk.
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, comman, 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 6).

<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>

Table 6. Event processor capacity in terms of maximum throughput (events per minute) comparison

The test was executed during a 240 x 2 jobs/min schedule for dynamic agents and fault tolerant agents. Figure 31 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 32). 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 31. Number of actions triggered by the event processor

Figure 32. 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 33. MDM CPU utilization comparison with and without event rule processing

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 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 (120 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.
4. Background Scheduling activities.

In the test environment, (with around network latency 0.1 ms), the file check throughput was strongly dependent on the scheduling workload. In this context the period between two files checks (in the total file dependencies list) passes from few hundredths of second (during no scheduling activity) a to tenth of second (during peak time) increasing the time to slide all the file dependencies every T (bm check file)

In a context like the actual one, it is suggested to tune the T (bm check file) with the following restriction:

\[
T > \frac{N}{10}
\]

being N the total number of file dependencies.
This scenario was applied to 2000 jobs scheduled with 2000 different file dependencies, for which \( T = \text{bm check file} = 300 \), as shown in the example above. 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 34).

![Throughput Dynamic Agent Scheduling (broker)](image)

**Figure 34.** File dependency releases in a 1200 jobs/min workload as baseline

### 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) bypassing the benefits 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.
4.1.5 Scheduling using “every” option

“Every” feature allows to create a new instance of a job stream or of a job in the plan, but while for the former case the impact is not relevant at run time, because the instances are already included in the plan, the latter causes multiple internal “sbj” to the same job stream instance that could affect performances especially in the plan updates.

It is very important to verify if the “EVERY” option is used with minimal values (few minutes) to avoid to increase the number of jobs in a job stream (few hundreds of job could impact performances). There are several methodologies that could help the driving of business need:

- Move EVERY option at job stream level if possible
- Split the Job stream in multiple job stream with “AT xx till xx” (time partitioning)

4.2 Dynamic domain manager table cleanup policy

While the workload increases in terms of the number of jobs executed per day on dynamic agents, 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 <DATA_DIR>/broker/config/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 7x10^5 rows of jobs that were 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 35. Plan update delay while dynamic job table is being cleaned up
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:

```
<INST_DIR>/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 `<DATA_DIR>/broker/config/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 dbuser -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 38. 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 10 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 39 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.
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 5 ms. Disk latency needs to be monitored constantly because if it increases over the desired threshold of 5 ms, the impacts on product performance would be tangible causing delays on real time scheduling.

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 – 10</th>
<th>10 -50</th>
<th>50 - 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 7. Dynamic Workload Console WebSphere Application Server Liberty Base heap configuration

<table>
<thead>
<tr>
<th>Schedule (jobs/min)</th>
<th>1 – 50</th>
<th>50 -100</th>
<th>100-200</th>
<th>&gt;200</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS Engine Heap size</td>
<td>2 GB</td>
<td>2.5 GB</td>
<td>4 GB</td>
<td>6 GB</td>
</tr>
</tbody>
</table>

Table 8. Engine WebSphere Application Server Liberty Base heap configuration

In addition to the above memory requirements, the native memory for the Java™ process and Workload Scheduler process should be taken into consideration.

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

As stated at the beginning of this document, starting from version 9.5, IBM Workload Scheduler has moved from WebSphere Application Server to WebSphere Application Server Liberty Base. As a result, the utilities known as wastools have been replaced with a number of templates addressing widely used configurations. You can now configure WebSphere Application Server Liberty Base to work with IBM Workload Scheduler using the templates provided or define your custom .xml files containing your own configuration settings.

WebSphere Application Server Liberty Base retrieves the .xml files from the overrides folder

- MDM/BKM/DDM: 
  `<TWA_DATA_DIR>/usr/servers/engineServer/configDropins/overrides`
- DWC: `<DWC_DATA_DIR>/usr/servers/dwcServer/configDropins/overrides`

and applies the configuration settings defined in each file. The file name is irrelevant, because WebSphere Application Server Liberty Base analyzes each .xml file for its contents.

By default, the datasource settings are specified into the datasource.xml file located in the overrides folder and these is the list of suggested values for both MDM and DWC nodes:

- statementCacheSize="400"
- isolationLevel="TRANSACTION_READ_COMMITTED"
- connectionTimeout="180s"
- maxPoolSize="300"
- minPoolSize="0"
- reapTime="180s"
- purgePolicy="EntirePool"

These are the values used to run all the workload scenarios described in this document.

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 jobs 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 = 70000`
- `com.ibm.tws.planner.monitor.cachemaxage = 21600000`

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 can be specified in the file on the Master Domain Manager:
`<DATA_DIR>/usr/servers/engineServer/resources/properties/TWSConfig.properties`. 
## 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 12.2.0.1.0 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><strong>WebSphere Application Server JVM max heap = min heap</strong></td>
<td>Required: 4096 for [10, 50] users per node Suggested: 6144 for [50, 150] users per node</td>
<td></td>
</tr>
<tr>
<td><strong>WebSphere Application Server JVM options</strong></td>
<td>-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><strong>WebSphere Application Server Liberty Base JDBC max Connections</strong></td>
<td>300</td>
<td>datasource.xml</td>
</tr>
<tr>
<td><strong>&lt;DATA_DIR&gt;/localopts</strong></td>
<td>bm check deadline = 0 bm check file = 120 bm check status = 300 bm check untils = 300 bm late every = 0 bm look = 5 bm read = 3 bm stats = off bm verbose = off</td>
<td>batchman settings</td>
</tr>
<tr>
<td><strong>Master Domain Manager</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WebSphere Application Server Liberty Base Configuration</strong></td>
<td>-Djava.awt.headless=true -Xdisableexplicitgc -Xgcpolicy:gencon –Xmn 1024m</td>
<td>-Xmn 1536m if heap size = 6144</td>
</tr>
<tr>
<td><strong>Heap</strong></td>
<td>Required: 4096 for [100, 200] jobs/min Suggested: 6144 for &gt;200 jobs/min</td>
<td></td>
</tr>
<tr>
<td><strong>Data Source</strong></td>
<td>JDBC Type = 4 Connection Timeout = 180 Max Connections = 300 Min Connections = 0 Purge Policy = EntirePool Reap Time = 180 Statement Cache Size = 400</td>
<td>datasource.xml</td>
</tr>
<tr>
<td><strong>DB (db2)</strong></td>
<td>LOGPRIMARY 200 LOGFILSIZ 3500 KEEPFENCED NO MAX_CONNECTIONS AUTOMATIC MAX_COORDAGENTS AUTOMATIC</td>
<td>MB total transaction log space</td>
</tr>
<tr>
<td>Setting</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STMT_CONC</td>
<td>LITERALS</td>
<td>This setting optimizes query execution and reduces CPU usage</td>
</tr>
<tr>
<td>SELF_TUNING_MEM</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>APPL_MEMORY, APPLHEAPSZ, DATABASE_MEMORY, DBHEAP, STAT_HEAP_SZ</td>
<td>AUTOMATIC</td>
<td></td>
</tr>
<tr>
<td>AUTO_RUNSTATS</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>AUTO_STMT_STATS</td>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>AUTO_REORG</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>PAGE_AGE_TRGT_MCR</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>TWS_PLN_BUFFPOOL</td>
<td>NPAGES</td>
<td>182000</td>
</tr>
<tr>
<td></td>
<td>PAGESIZE</td>
<td>4096</td>
</tr>
<tr>
<td>TWS_BUFFPOOL_TEMP</td>
<td>NPAGES</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>PAGESIZE</td>
<td>16384</td>
</tr>
<tr>
<td>TWS_BUFFPOOL</td>
<td>NPAGES</td>
<td>50000</td>
</tr>
<tr>
<td></td>
<td>PAGESIZE</td>
<td>8192</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Directory</th>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/broker/config/JobDispatcherConfig.properties</td>
<td></td>
<td>Historical data management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MoveHistoryDataFrequencyInMins=720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SuccessfulJobsMaxAge = 360</td>
</tr>
<tr>
<td>/broker/config/ResourceAdvisorConfig.properties</td>
<td></td>
<td>MaxAllocsPerTimeSlot = 1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TimeSlotLength = 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MaxAllocsInCache = 50000</td>
</tr>
<tr>
<td>/usr/servers/engineServer/resources/properties/TWSConfig.properties</td>
<td></td>
<td>Plan replication configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>com.ibm.tws.planner.monitor.subProcessors = 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>com.ibm.tws.planner.monitor.filecachesize= 40000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>com.ibm.tws.planner.monitor.cachesize = 70000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>com.ibm.tws.planner.monitor.cachemaxage = 21600000</td>
</tr>
</tbody>
</table>

Table 9. Main configurations and tunings

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 9.5.0.x 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><strong>10K jobs (50% FTA +50% DYN) per day (8 jobs/min), 10 concurrent users</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Node</td>
<td>WS Engine, RDBMS, DWC</td>
<td>4</td>
</tr>
<tr>
<td><strong>250K jobs (50% FTA +50% DYN) per day (175 jobs/min), 50 concurrent users</strong></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><strong>500K jobs (50% FTA +50% DYN) per day (350 jobs/min), 100+ concurrent users</strong></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 10. Capacity planning samples
# 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.

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 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 150,000 ideapreneurs are in a relationship Beyond the Contract™ with 500+ customers in 46 countries. how can I help you?