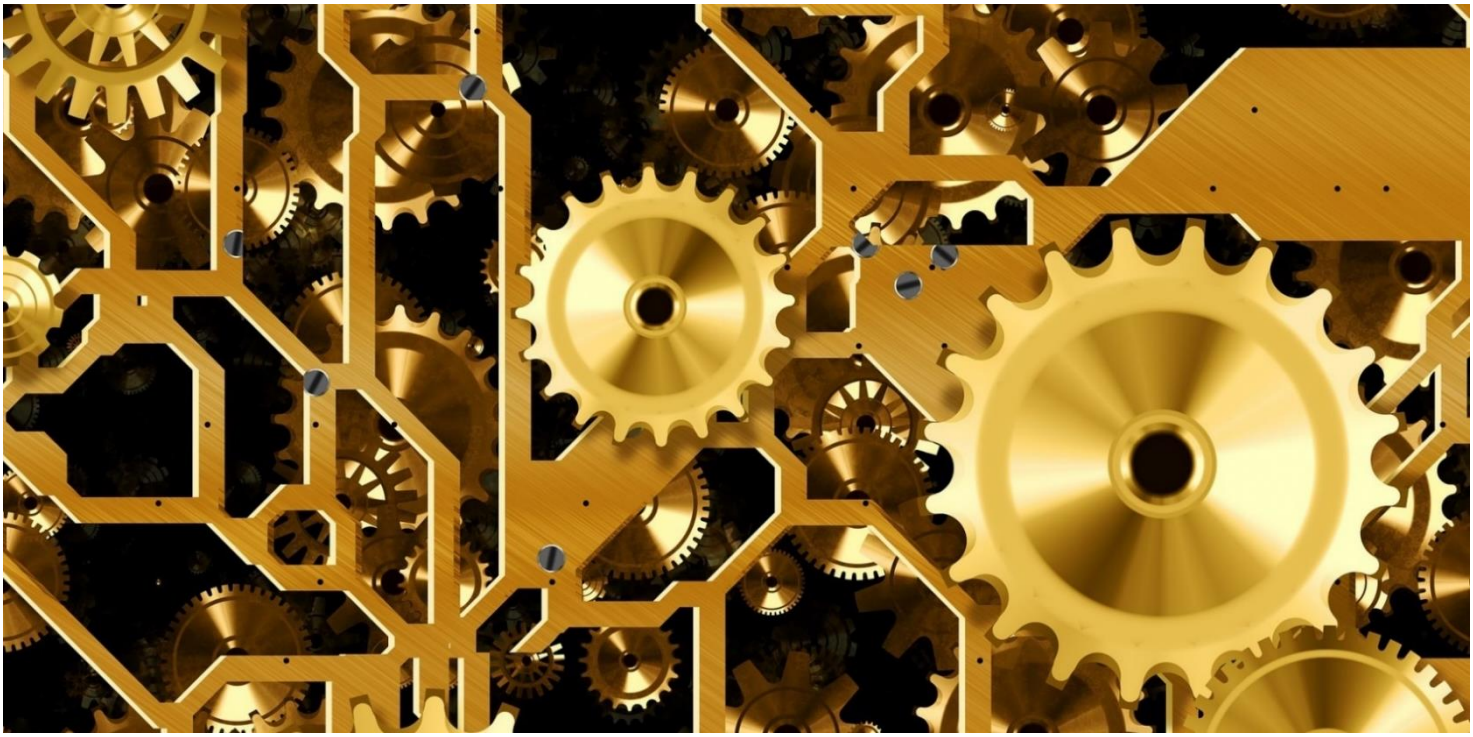

IBM Workload Scheduler
HCL Workload Automation

V10.2 Performance Report

Document version 1.0

Lorenzo Nichele
Paolo Cavazza

Workload Automation Performance Team - HCL Software Rome Lab

HCLSoftware

HCLTech

This edition applies to version 10, release 2, fix pack 0 of IBM Workload Scheduler (IWS) and HCL Workload Automation (HWA) and to all subsequent fix packs and modifications unless otherwise stated in new editions.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

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.

For more details about the new features introduced with version 10.2, see the "Summary of enhancements" section in the online product documentation:

IBM Workload Scheduler version 10.2.0 enhancements
HCL Workload Automation version 10.2.0 enhancements

2 Executive Summary

The objective of the tests described in this document is to report the performance results for the new version of the product, 10.2, in comparison with previous versions (see Workload Scheduler 9.5.0.2 performance report and Workload Scheduler 10.1.0.1 Performance Report).

The test results can be summarized as follows:

- **Scheduling delay** for job submissions to dynamic agents: for scheduling workloads up to 1400 jobs/min, version 10.2 showed the same average delays as previous versions (between 30 and 40 seconds). For peak workloads of around 5000 jobs/min, only the test environment with a higher number of job streams in plan (around 43,000) exhibited average delays of around 60 seconds, a ~20% increase compared to version 10.1, while the test environment with fewer job streams in plan (around 10,000) presented the same average delays as previous versions (around 40 seconds).
- Jobs and job streams status update delay for Dynamic Workload Console ("**mirroring delay**"): both test environments showed the same mirroring delays as previous versions (average delays at most between 30 and 40 seconds) for all levels of the scheduling workload.
- **CPU utilization**:
    - For the test environment with around 43,000 job streams in plan, average CPU utilization at max load (~5000 jobs/min) showed an increase in the range 15%-20% compared to version 10.1, for both MDM and DB machines.
    - For the test environment with "encryption at rest" enabled and around 10,000 job streams in plan, an increase of about 45% in average CPU utilization at max load (~5000 jobs/min) for the MDM machine has been observed compared to version 10.1, while the DB machine showed the same utilization as previous versions
- **Disk utilization**: test measurements confirmed that the disk on the MDM system is the most utilized resource, as for previous versions, with average values in the range 90%-100% at max load (~5000 jobs/min).

- For the test environment with around 43,000 job streams in plan, compared to version 10.1, average disk utilization on the MDM system at max load (~5000 jobs/min) increased by nearly 10%, while for the DB system, the increase was almost 40%.
- For the test environment with “encryption at rest” enabled and around 10,000 job streams in plan, compared to version 10.1, average disk utilization increased by nearly 10% for both the MDM and the DB system at max load (~5000 jobs/min).

3 Performance Test

3.1 Test Approach

The focus of performance tests for Workload Scheduler 10.2 was to validate that the new version of the product yields the same results, in terms of throughput and processing times, of previous releases.

In this context, continuous monitoring of key performance indicators (scheduling throughput, mirroring and scheduling average delays) and hardware resources (CPU, disk) has been implemented to measure product performance during long run scenarios.

3.2 Environment

The test environments were 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 Gbit/s VLAN.

Two test environments were used, the main difference being the database server: DB2 for one test environment, Oracle for the other.

The following table summarizes operating system and middleware versions:

OS	Red Hat Enterprise Linux release 9.0 Kernel 5.14.0-70.30.1.el9_0.x86_64	
Database	IBM DB2 v11.5.8	Oracle® 19c Enterprise Edition
Application Server	Liberty 23.0.0.6 IBM Semeru Runtime Open Edition (11.0.19+7)	

Table 1. Software level of code

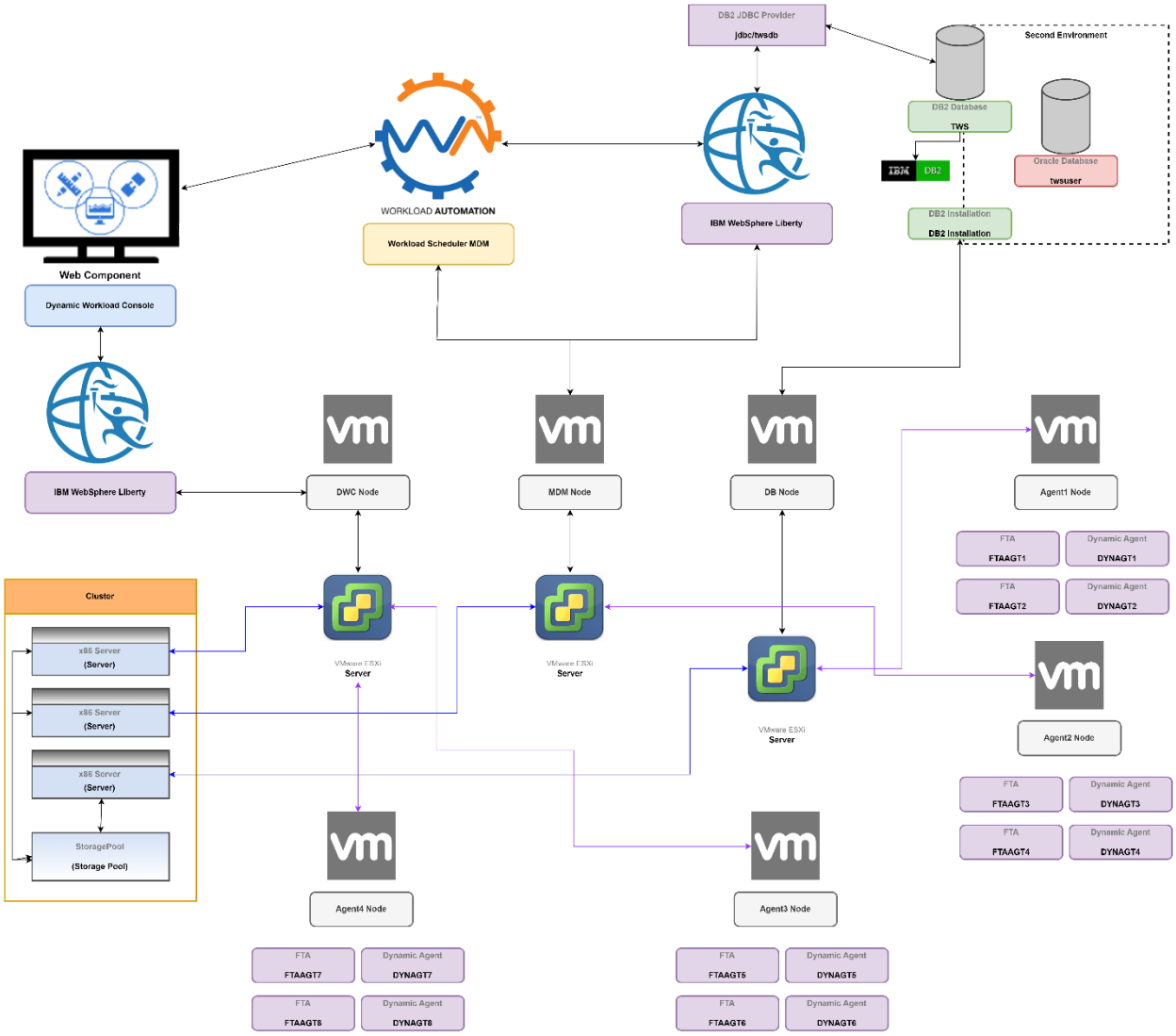

Figure 1. Overall deployment view of test environment

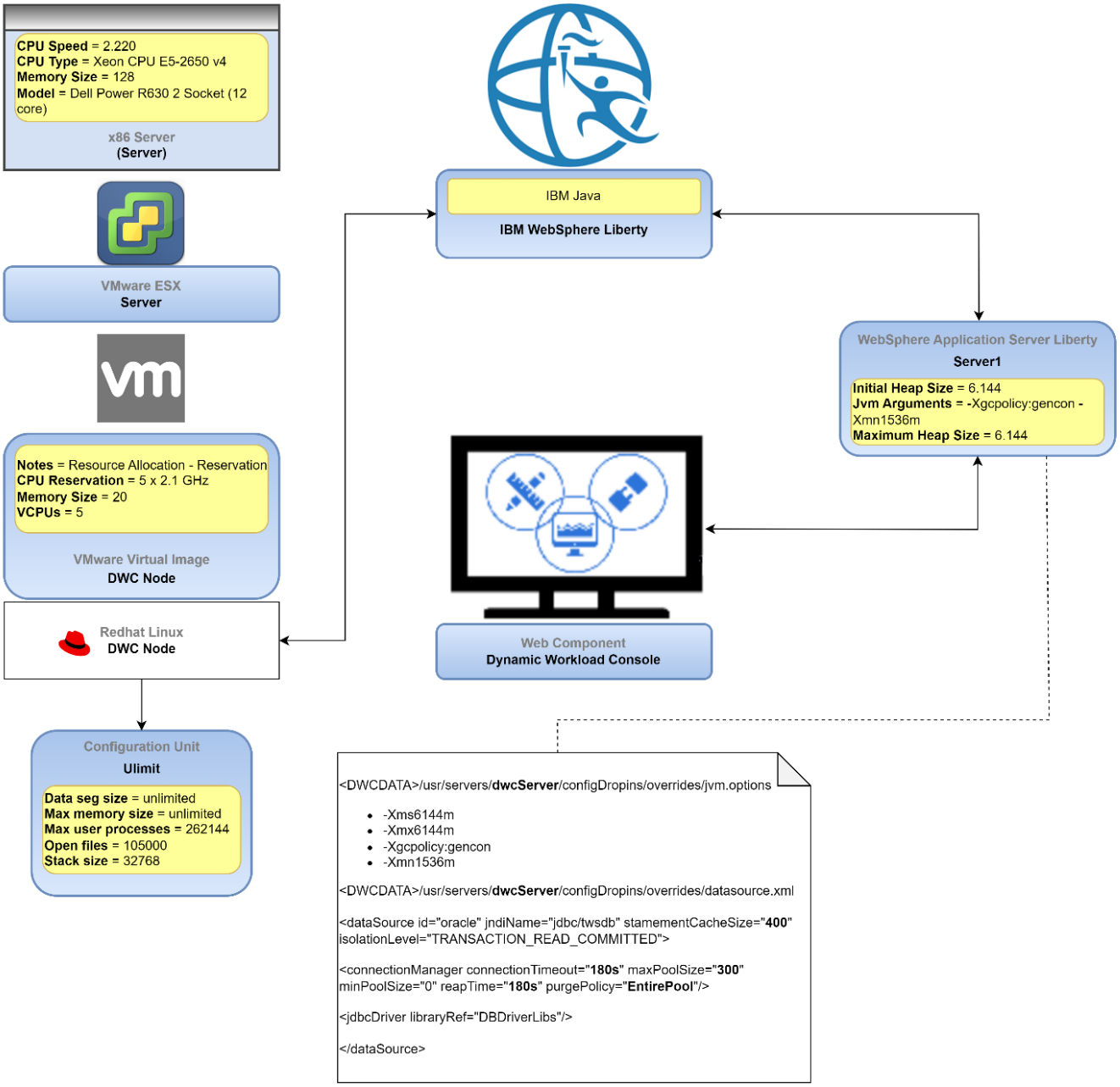

Figure 2. Dynamic Workload Console node configuration

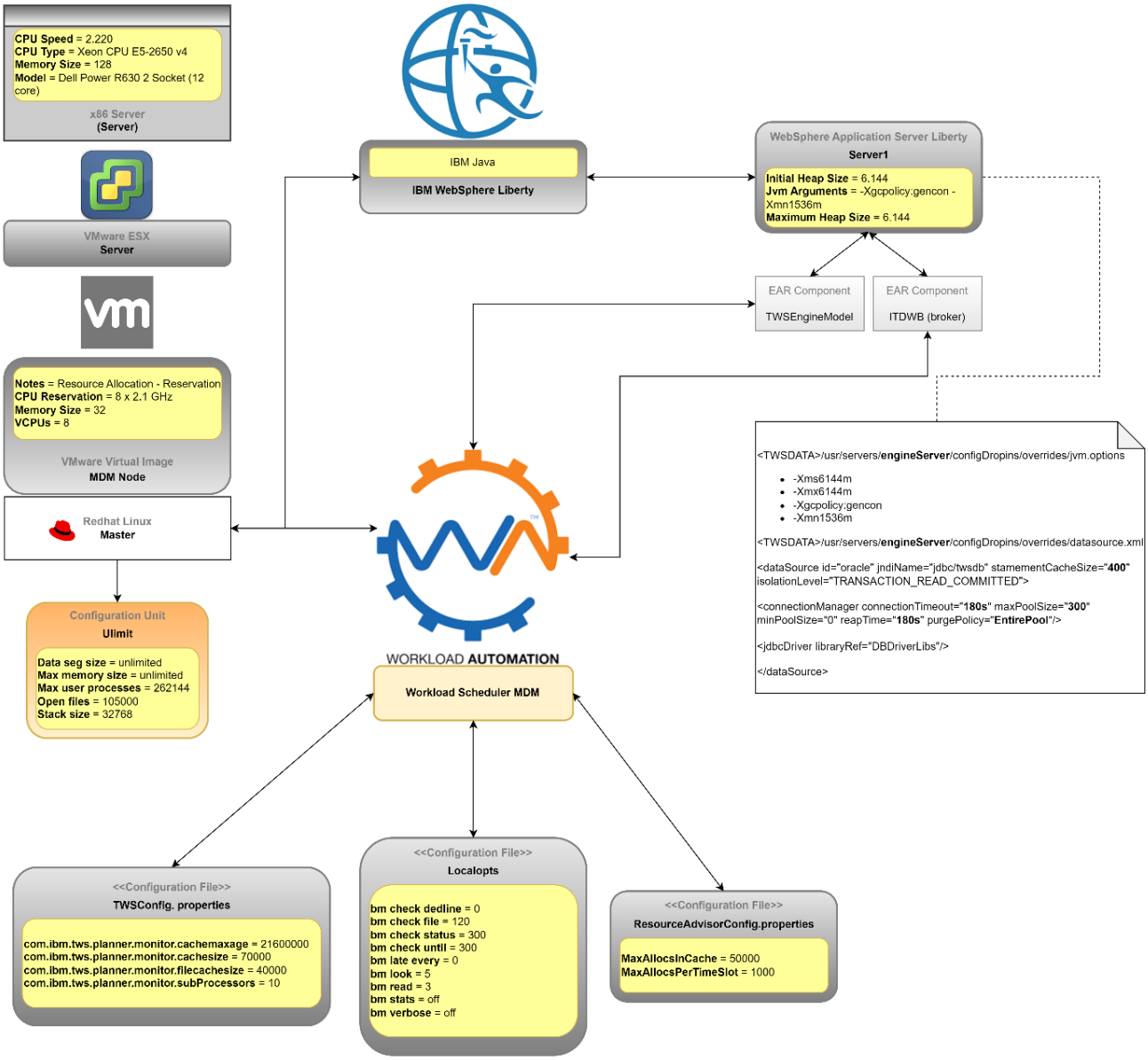

Figure 3. Master Domain Manager node configurations

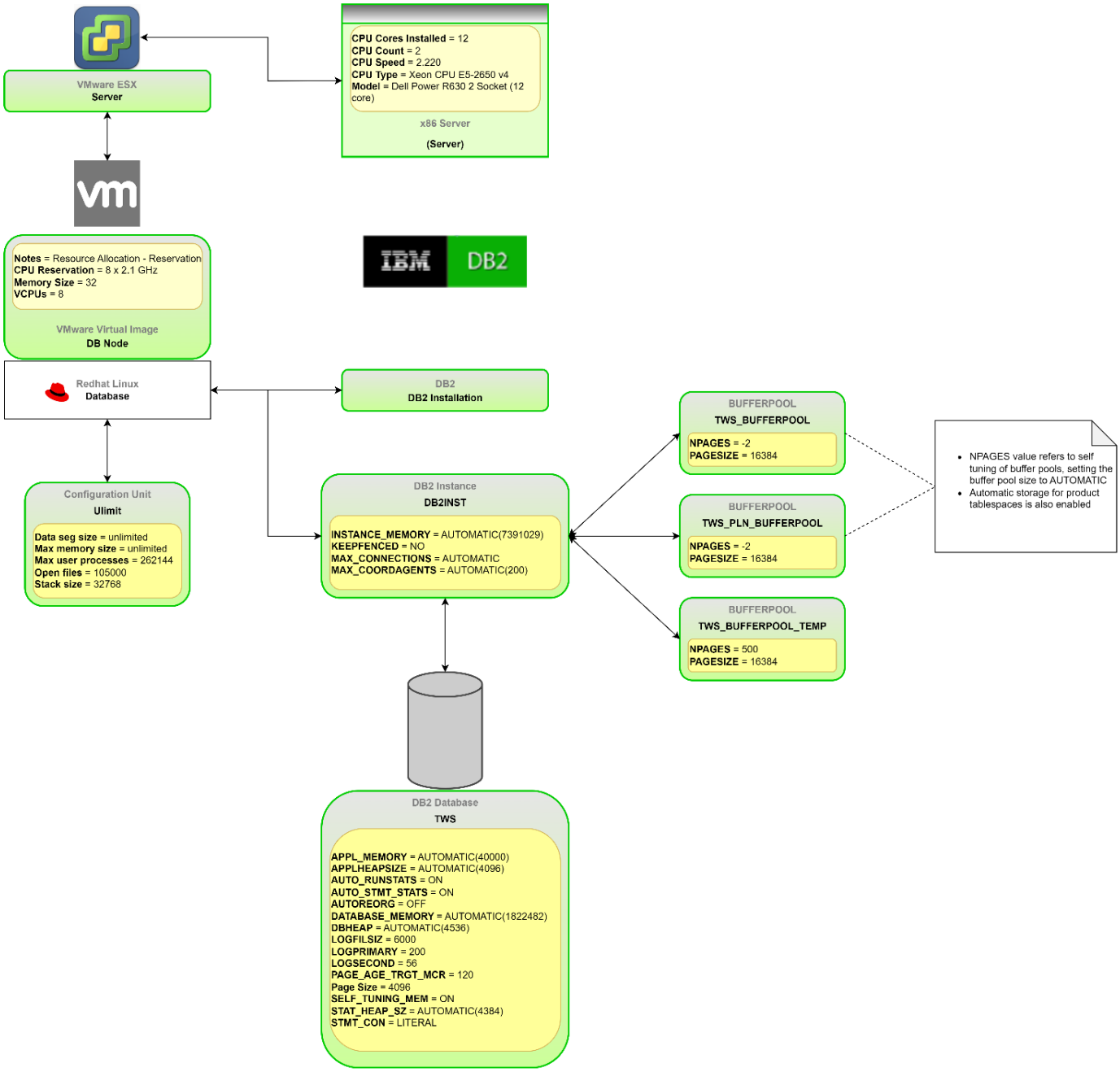

Figure 4. Database node configurations

Regarding the DB system, the same configuration (CPU reservation: 8 x 2.1 GHz; RAM reservation: 32GB) has been applied to the Virtual Machine hosting Oracle 19c Server without any further customization related to Oracle database.

Figure 5 shows the storage (disk) configuration, which is a critical detail of the test environments because disk is the most utilized hardware resource.

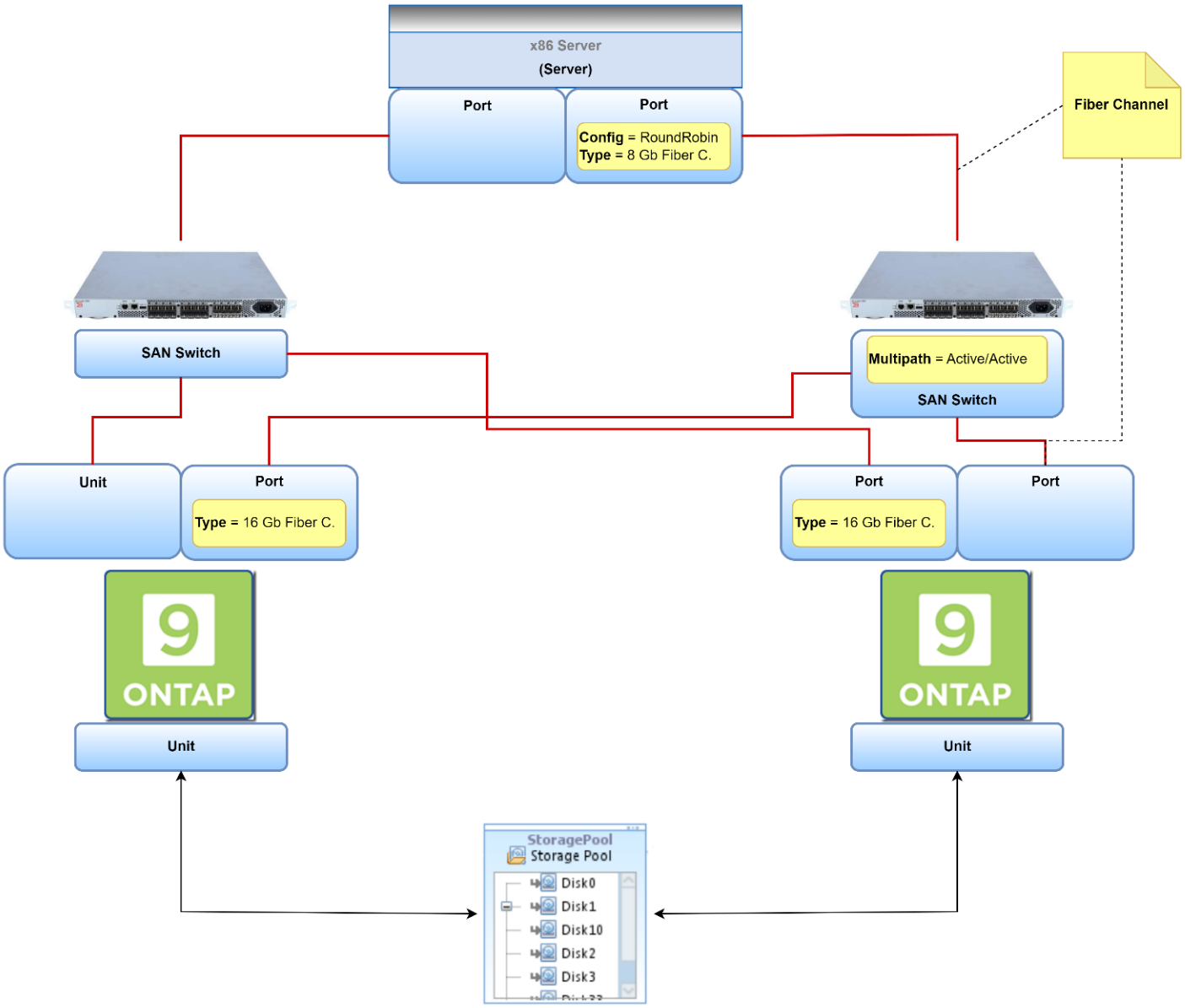

Figure 5. Storage Solution

3.3 Test Workload and Results

3.3.1 Scheduling Workload

This section reports the details of the workload included in the production plan executed daily in the Performance Test environments. The workload is distributed among several fault tolerant and dynamic agents. Each day, the total number of job streams that are executed is around 10,000 for the test environment with Oracle as database, and around 43,000 for the test environment with DB2 as database. In terms of jobs, around 620,000 are executed daily for the test environment with Oracle as database, while around 530,000 are executed daily for the test environment with DB2 as database.

SSL Communication across the fault-tolerant agent network has been enabled for both test environments.

Encryption at rest for key product files, such as the `Symphony` file, messages queues, and the `useropts` file, has been enabled for the test environment with Oracle as database.

Scheduling objects (such as jobs, job steams, and workstations) have been organized in a hierarchy of folders, around 30 for the test environment with DB2 as database, and around 400 for the test environment with Oracle as a database.

Standard FTA and Dynamic Agent schedule
• The plan includes **124800 jobs scheduled in around 3 hours**. There are **52000 jobs** scheduled to be executed during a 10-minute peak. In addition, only for dynamic agents, around **361000** jobs are scheduled in 5 hours (1200 jobs/min).
WSA - Critical path
• 48 complex patterns, composed of 4 linked job streams with 10 jobs each. 4 jobs for each complex pattern are defined as critical jobs
EDWA
• **200 TWS-Objects rules** - each rule matches a workstation and job name belonging to the daily production plan and the success state of job execution. In case of event matching, the action is to create a new message log. Normally, 4140 events (Message loggers) are generated at the end of each test run. • **File-created rules** -These event monitor rules generate a specific message logger each time a new file with a predefined naming convention is created on each agent. In total, 240 events (message loggers) were generated each hour, which means 1 event every 4 minutes on each of the 16 agents.
Conditional Dependencies
• 5% of additional workload, an additional 3200 jobs/800 job streams over 4 dynamic agents and 4 FTA. This means that there are 100 job streams for each agent, half of which have internal dependencies and the other half has external dependencies. In the case of conditional dependencies, there are also 800 join conditions overall.
Ad Hoc Submission (conman sbs)
• Dynamic submission of jobs using the command "conman sbs" to submit a job stream with 20 different jobs (5 per agent) with dependencies between them in a chain. In total, 1000 dynamic jobs were submitted in 10 minutes, from 16:40 to 16:50.
File Dependencies
• 35000 job streams with a single job definition and a single file dependency defined at job stream level distributed across the 8 FTAs in the test environment (4375 jobs per agent). These 35000 job streams have a time dependency that is different for each FTA: the single block of 4375 job streams per agent is scheduled to start one hour after the preceding one for a duration of 8 hours long.
File Transfer
• **SSH protocol** - 1 File Transfer job scheduled to run every hour from 16:15 until 20:15 (5 instances per day). This File Transfer job is configured to copy 10 files x 1GB size from a SSH Server to a dynamic agent. • **Workstation-to-Workstation protocol -** 3 File Transfer jobs configured to copy from a dynamic agent to another dynamic agent respectively 5 files x 1GB size (6 instances per day starting from 04:00 until 04:55, one instance each 11 minutes); 50 files x 100KB size (6 instances per day starting from 06:00 until 06:55, one instance each 11 minutes); 100 files x 1MB size (6 instances per day starting from 08:00 until 08:55, one instance each 11 minutes).

Table 2. Daily plan workload composition

This workload is used as a benchmark to track changes in key performance indicators over time and across product versions.

3.3.2 Results

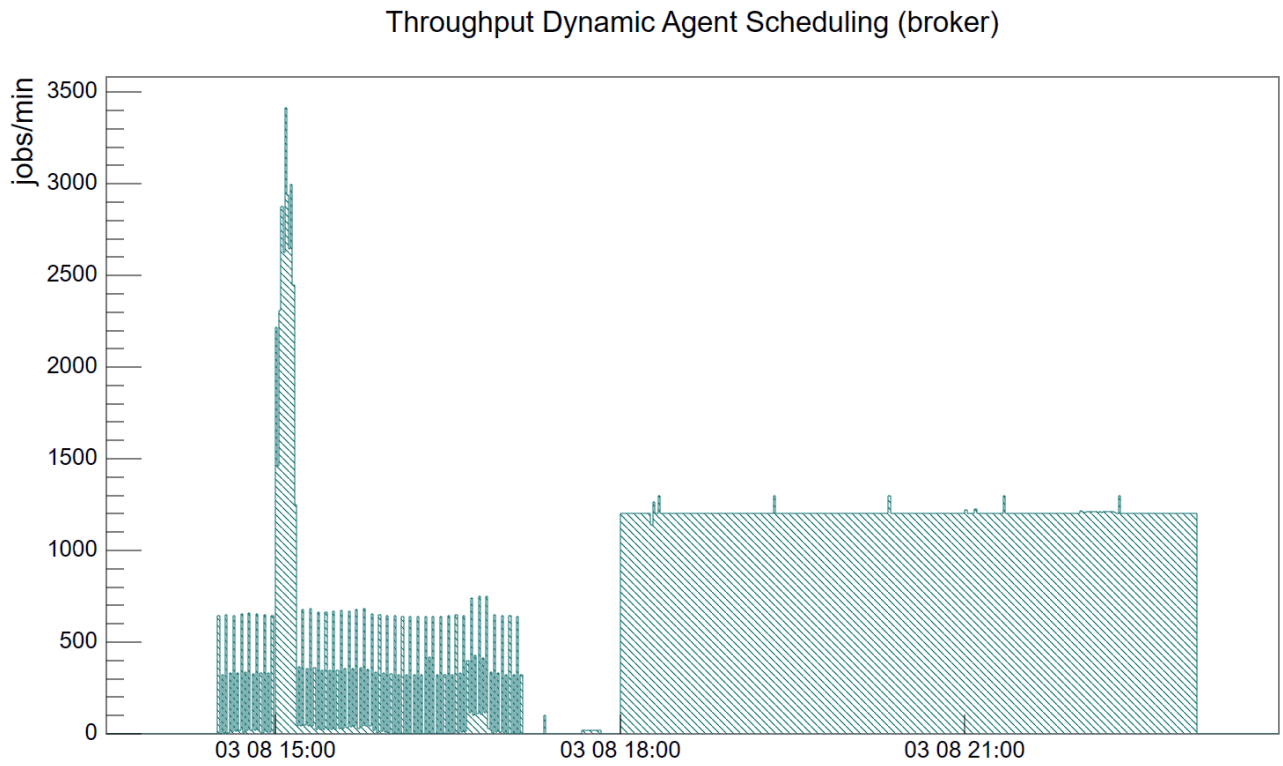

Figure 6. Dynamic agent throughput in daily workload

Figure 6 shows the daily workload (in terms of jobs executed per minute) of jobs scheduled on dynamic agents. In the time range between 18:00 and 23:00, jobs are executed by means of job streams with the “every” option.

Figure 7 shows the daily workload of jobs scheduled on Fault Tolerant Agents.

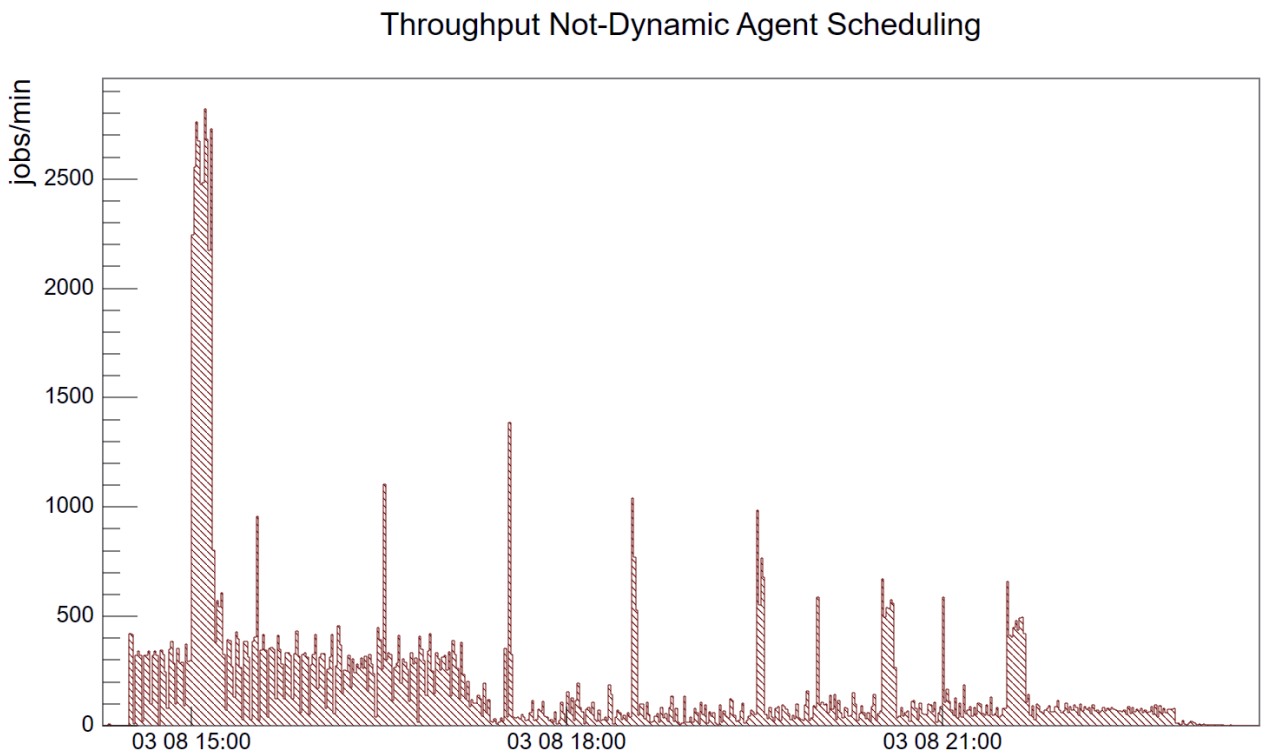

Figure 7. Fault tolerant agent schedule throughput in daily workload

The scheduling throughput described in this section is the same observed in previous releases.

In terms of scheduling delay for job submissions to dynamic agents, the average of measured values was consistently below 60 seconds, except for the time range when the maximum workload (more than 5,000 jobs per minute) is applied to the system (see Figure 8)

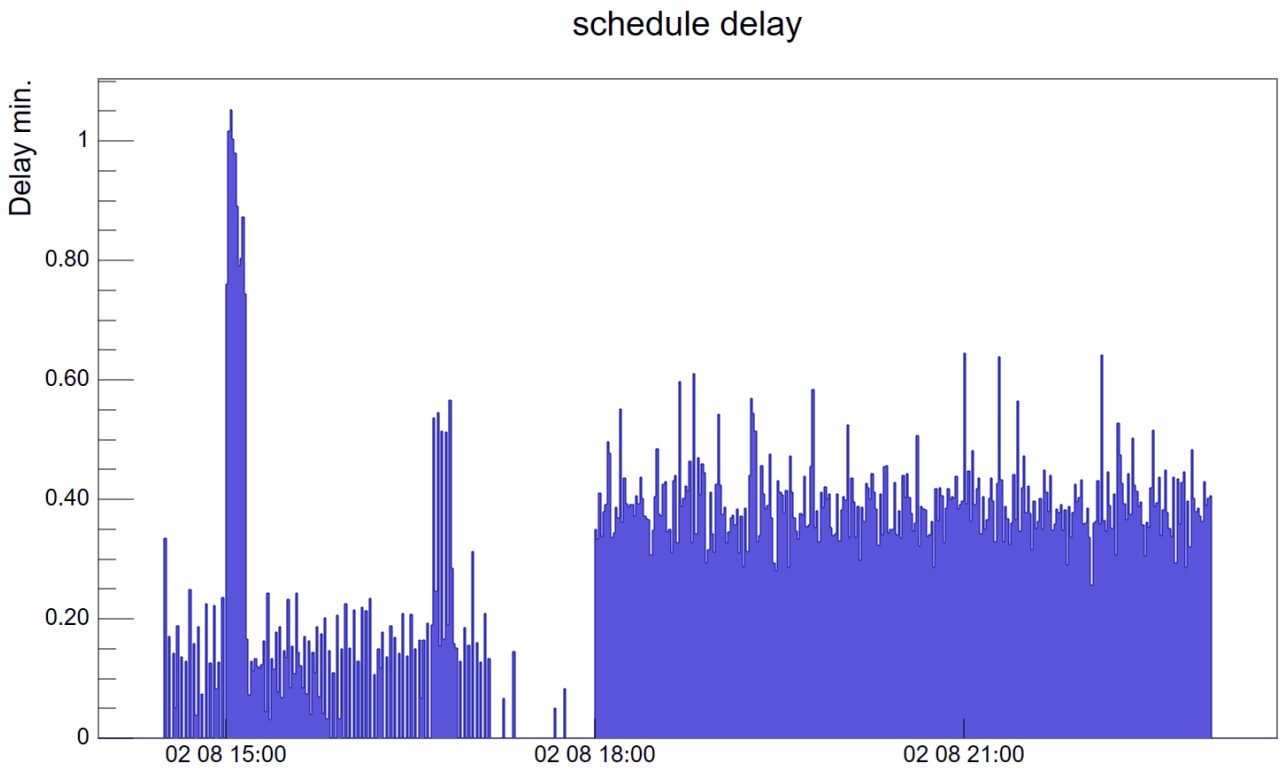

Figure 8. Average job schedule delay per minute over time (jobs scheduled on dynamic agents)

Regarding jobs and job streams status update delay for Dynamic Workload Console ("mirroring delay"), the average of measured values was consistently below 60 seconds for all workload levels applied to the test environments (see Figure 9).

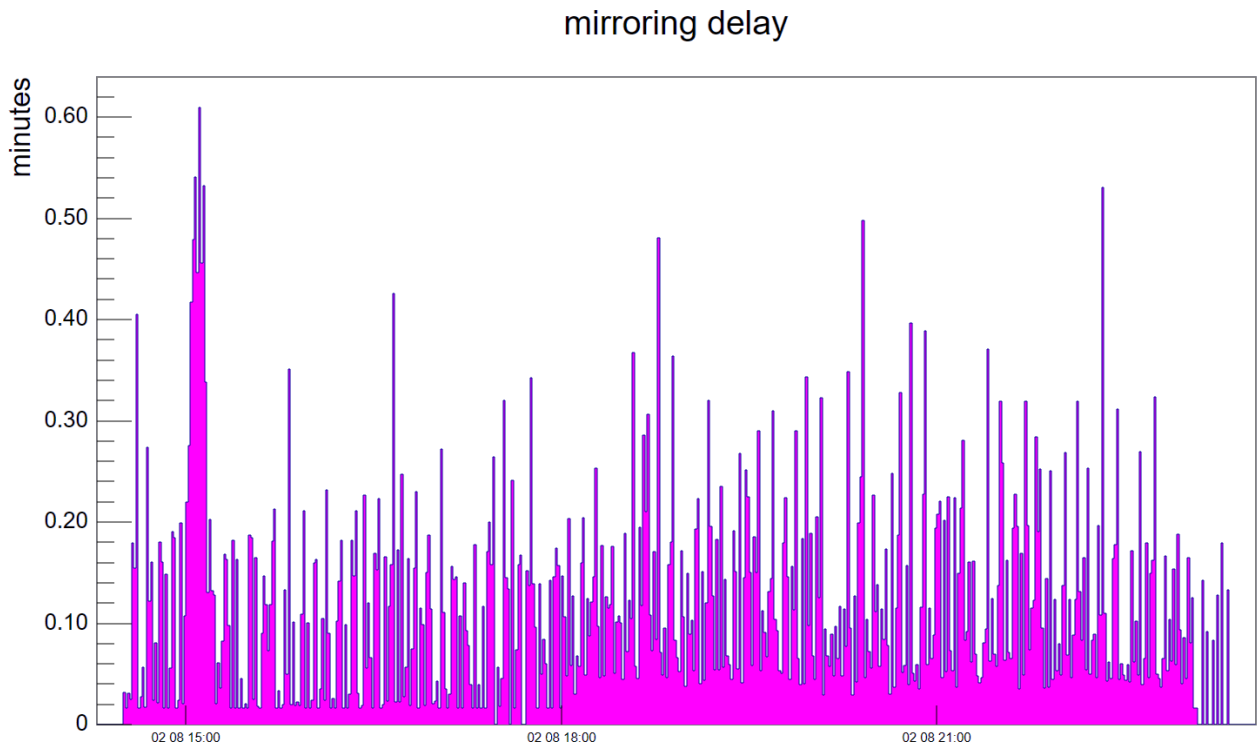

Figure 9. Job plan status update delay over time

These results are strictly related to the test environment and workload described in previous sections; the same results could not be achieved using a different environment and workload.

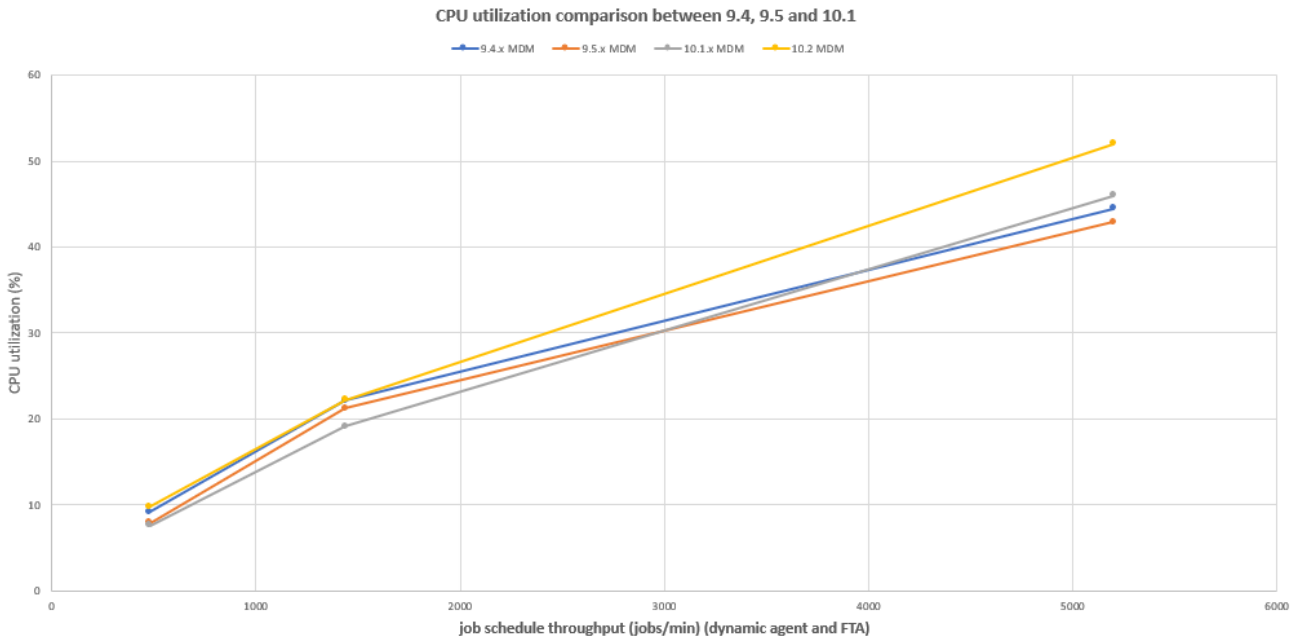

Figure 10. Total average CPU utilization at Master Domain Manager vs different workload

Figure 10 shows CPU utilization on the MDM machine in relation to the total (both dynamic and fault tolerant agents) scheduling throughput as described in section 3.3.1. Version 10.2 showed an increase of 15%-20% in

CPU utilization at max load (~5,200 jobs/min).

As for previous versions, also for 10.2, the disk on the MDM system is the most utilized HW resource. Figure 11 shows that, during peak load (more than 5000 jobs/min, time frame 15:00 – 15:10), average disk utilization on the MDM system is above 90%.

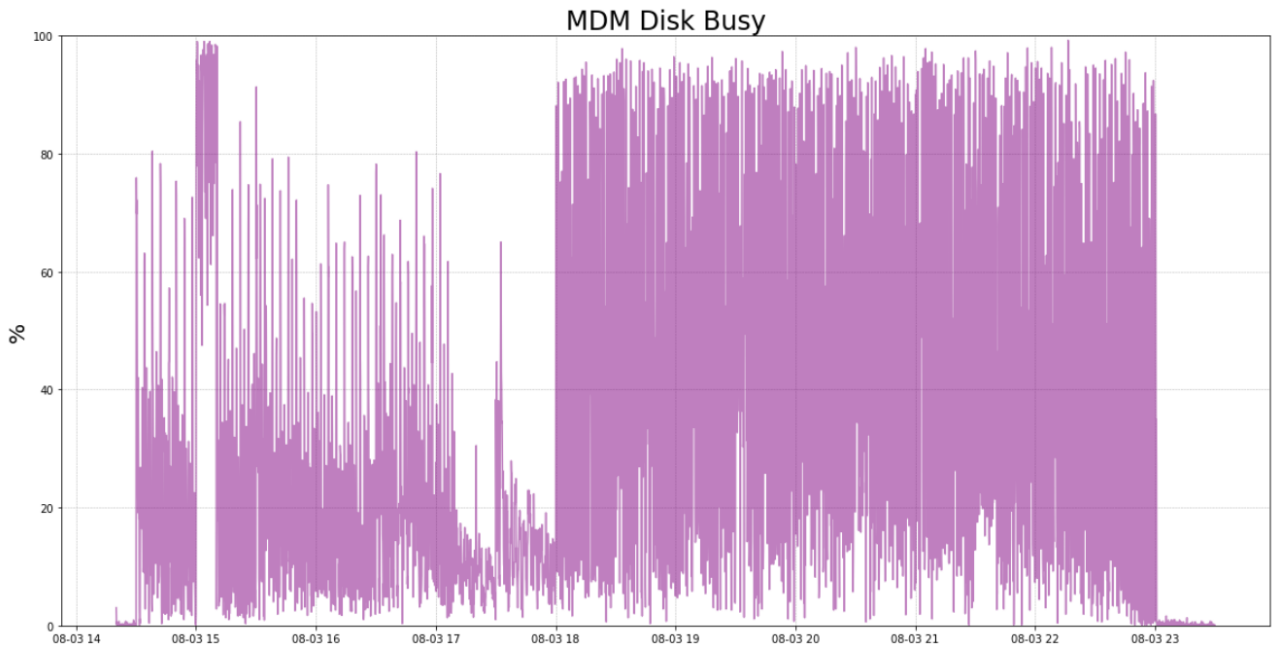

Figure 11. Disk utilization of the Master Domain Manager

For the test environment with DB2 as a database, a two-disk setup has been implemented, configuring DB2 to use one disk for the actual data and a second disk for the transaction logs.

Figure 12 shows an average utilization between 50% and 60%, at max load (~5,000 jobs/min), for the disk holding the data.

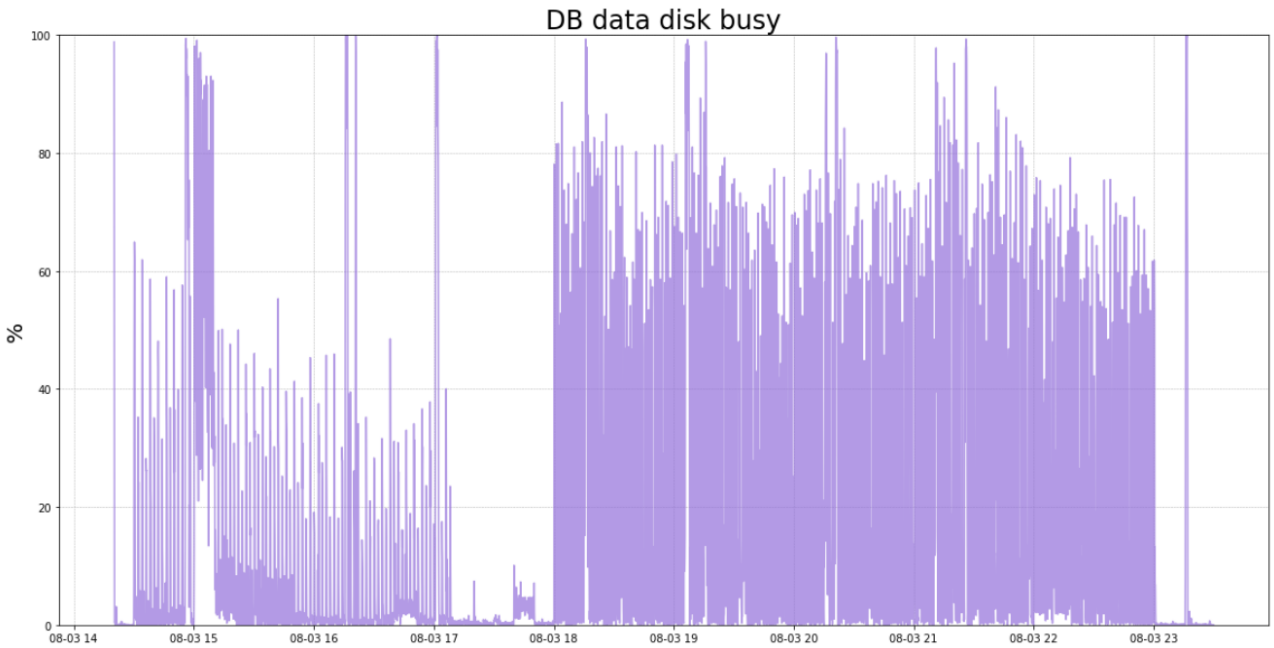

Figure 12. Database data disk utilization

Figure 13 shows an average utilization between 70% and 80%, at max load (~5,000 jobs/min), for the disk dedicated to the transaction logs.

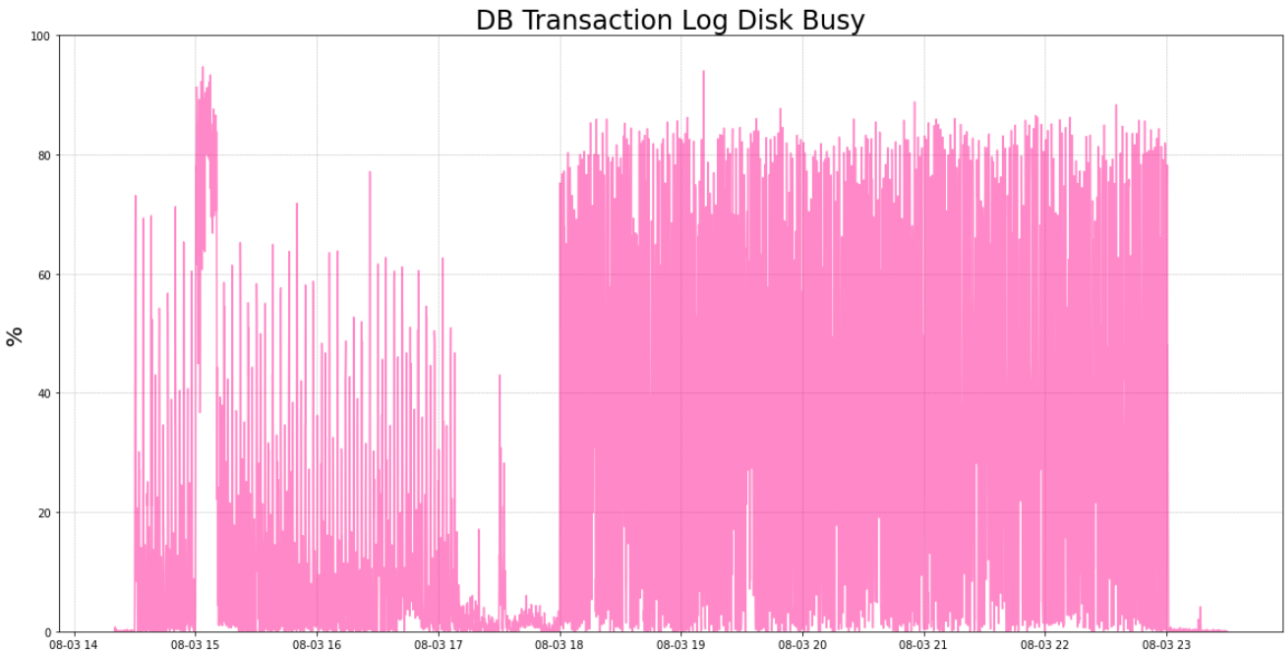

Figure 13. Database transaction log disk usage

4 Best Practices

4.1 Scheduling

Workload Automation software offers many features to perform at best its own objective: orchestrate scheduling flow activities. During the plan generation activity, the principal way to schedule is to have job streams associated with a run cycle and include them in the plan. In addition, the schedule of jobs and job streams could occur dynamically while the plan is running, for example, using event rules, conman, start conditions, and file dependencies. Even if the latter gives a higher level of versatility to accomplish different business scenarios, some recommendations must be considered before planning to adopt file dependencies to orchestrate the schedule in case of a heavy high 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 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 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 horizontally but vertically only by increasing the CPU and/or I/O capabilities.

Detailed considerations about Event Processor Throughput capabilities are available in the 4.1.1 section of the Workload Scheduler 9.5.0.2 performance report.

4.1.2 Scheduling using file dependencies

Workload Automation 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 3.3.1 already includes a component of this type acting against fault tolerant agents. This feature has a different impact on the performance if used with a dynamic agent. In the case of a dynamic agent, the entire mechanism is driven by the dynamic domain manager that oversees 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 the server about file status. The server workload throughput is ruled by four parameters:

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

In the test environment 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) 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.

A practical example of the considerations above can be found in the 4.1.2 section of the Workload Scheduler 9.5.0.2 performance report.

4.1.3 Scheduling using conman sbs

"`conman sbs`" command (or equivalent REST API calls) 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 several hundred objects, considering both jobs in the job streams and internal and/or external dependencies.

4.1.4 Scheduling using "every" option

"Every" feature allows to create a new instance of a job stream or 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 especially important to verify if the “EVERY” option is used with minimal values (a few minutes) to avoid increasing the number of jobs in a job stream (a few hundred jobs could impact performances). Several methodologies could help to drive the business needs:

- 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 performing job log retrieval for archived plans. The following parameters, in the `<TWA_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 boots and removes jobs from the table according to 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 Automation performance and throughput capabilities. For more details about negative performance impacts, refer to 4.2 section of the Workload Scheduler 9.5.0.2 performance report.

To avoid the behavior described above, it could be suggested to manage the policy in a more controlled way. For instance, a specific job could be used to run the cleanup invoking the built-in Workload Automation 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, as shown in the example below, a job stream named “CLEANUP_DWB_DB” which invokes the script `movehistorydata.sh` is scheduled to run every day at 23:45:

```
SCHEDULE MDMWS#CLEANUP_DWB_DB
DESCRIPTION "Added by composer."
ON RUNCYCLE RULE1 "FREQ=DAILY;INTERVAL=1"
AT 2345
:
MDMWS#CLEANUP_DWB_DB
 SCRIPTNAME "<MDM_INST_DIR>/TDWB/bin/movehistorydata.sh -dbUsr db2user -dbPwd
xxxxxxx -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
```

If this suggested implementation is adopted, it is needed to properly configure the file `<TWA_DATA_DIR>/broker/config/JobDispatcherConfig.properties`.

In particular, the values specified for `SuccessfulJobsMaxAge` and `MoveHistoryDataFrequencyInMins` in the `JobDispatcherConfig.properties` file need to be higher than the values specified in the MDMWS#CLEANUP_DWB_DB job (see example above):

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

5 Recommendations

5.1 CPU

All tests described in this document were executed on virtual CPUs assigned exclusively to VMs (reserved resources). Under this assumption, the information provided in Table 8 could be a starting point to figure out the hardware required to support a given workload.

5.2 Storage

Disk performance (latency, throughput) is critical for the impact it can have on scheduling delay. Results described in this report were achieved using the storage configuration described in Figure 5.

5.3 Memory

For the MDM, the suggested configuration of the JVM heap size in relation with a desired scheduling throughput (jobs executed per minute) can be found in the following table:

Scheduling throughput (jobs/min)	1 – 50	50 – 100	100 – 200	>200
WS Engine Heap size	2 GB	2.5 GB	4 GB	6 GB

Table 3. Engine Liberty JVM heap configuration

In addition to the above memory requirements, the native memory for the Java™ process and Workload Automation process should be taken into consideration. This means that the RAM of the machines where the IBM WebSphere Application Server Liberty is running needs to be between 50% and 100% larger than the JVM max heap size.

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

You can configure Liberty to work with Workload Automation using the templates provided or defining your custom .xml files containing the customized configuration settings.

Liberty 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 Liberty analyzes each .xml file for its contents.

By default, the data source 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 through parallelism (multithreading) and caching. The former has 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 configurations (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 managed:

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

Since Workload Automation 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:
`<TWA_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 19c Enterprise Edition 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 or greater.

5.4.4 Comprehensive configuration and tuning

Dynamic Workload Console		
Component	**Settings**	**Comment**
Liberty JVM options	-Xms<heap size> -Xmx<heap size> -Xmn<nursery size> -Xgcpolicy:gencon -Xdisableexplicitgc	Heap size: • 4096m for up to 50 users/instance • 6144m for up to 150 users/instance Nursery size: ¼ of heap size Configuration file: jvm.options in <DWC_DATA_DIR>/usr/servers/dwcServer/configDropins/overrides/
Liberty Datasource	JDBC max Connections = 300	Configuration file: datasource.xml in <DWC_DATA_DIR>/usr/servers/dwcServer/configDropins/overrides/

Table 4. DWC recommended settings.

Master Domain Manager		
Component	Settings	**Comment**
batchman	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	Configuration file: <TWA_DATA_DIR>/localopts
Liberty JVM options	-Xms<heap size> -Xmx<heap size> -Xmn<nursery size> -Xgcpolicy:gencon -Xdisableexplicitgc	Heap size: • 4096m for up to 200 jobs/min • 6144m for more than 200 jobs/min Nursery size: ¼ of heap size Configuration file: jvm.options in <DATA_DIR>/usr/servers/dwcServer/configDropins/overrides/
Liberty Datasource	JDBC Type = 4 Connection Timeout = 180 Max Connections = 300 Min Connections = 0 Purge Policy = EntirePool Reap Time = 180 Statement Cache Size= 400	Configuration file: datasource.xml in <DATA_DIR>/usr/servers/dwcServer/configDropins/overrides/

Table 5. MDM recommended settings.

DB2			
Parameter		Value	**Comment**
LOGPRIMARY		200	**LOGPRIMARY**: total number of transaction logs **LOGFILSIZ**: number of 4KB pages for each transaction log **Total transaction logs size**: 200 * 6000 * 4 ≈ 4.6 GiB
LOGFILSIZ		6000	
KEEPFENCED		NO	
MAX_CONNECTIONS		AUTOMATIC	
MAX_COORDAGENTS		AUTOMATIC	
STMT_CONC		LITERALS	This setting optimizes query execution and reduces CPU usage
SELF_TUNING_MEM		ON	
APPL_MEMORY **APPLHEAPSZ** **DATABASE_MEMORY** **DBHEAP** **STAT_HEAP_SZ**		AUTOMATIC	
AUTO_RUNSTATS		ON	
AUTO_STMT_STATS		ON	
AUTO_REORG		OFF	
PAGE_AGE_TRGT_MCR		120	
TWS_PLN_BUFFPOOL	NPAGES	-2	Automatic (self-tuning)
	PAGESIZE	16384	
TWS_BUFFPOOL_TEMP	NPAGES	500	
	PAGESIZE	16384	
TWS_BUFFPOOL	NPAGES	-2	Automatic (self-tuning)
	PAGESIZE	16384	

Table 6. DB2 recommended settings.

Dynamic Workload Broker		
Feature	**Settings**	**Comment**
Historical data management	MoveHistoryDataFrequencyInMins = 720 SuccessfulJobsMaxAge = 360	Configuration file: JobDispatcherConfig.properties in <TWA_DATA_DIR>/broker/config/
Dynamic scheduling	MaxAllocsPerTimeSlot = 1000 TimeSlotLength = 10 MaxAllocsInCache = 50000	Configuration file: ResourceAdvisorConfig.properties in <TWA_DATA_DIR>/broker/config/
Plan replication configuration	com.ibm.tws.planner.monitor.subProcessors = 10 com.ibm.tws.planner.monitor.filecachesize = 40000 com.ibm.tws.planner.monitor.cachesize = 70000 com.ibm.tws.planner.monitor.cachemaxage = 21600000	Configuration file: TWSConfig.properties in <TWA_DATA_DIR>/usr/servers/engineServer/resources/properties/

Table 7. Dynamic Workload Broker recommended settings.

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 DWC 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 version 10.1.0.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 cores.

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.

	NODE	Number of vCPUs	RAM (GB)
up to 10K jobs (50% FTA +50% DYN) per day (~8 jobs/min), 10 DWC users			
1 Node	**WS Engine, RDBMS, DWC**	4	16
up to 100K jobs (50% FTA +50% DYN) per day (~70 jobs/min), 50 DWC users			
2 Nodes	**WS Engine, DWC**	4	16
	RDBMS	4	16
up to 600K jobs (50% FTA +50% DYN) per day (~410 jobs/min), 100+ DWC users			
3 Nodes	**WS-Engine**	8	32
	RDBMS	8	32
	DWC	5	20

Table 8. Capacity planning examples

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 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.

NETAPP, the NETAPP logo, and the marks listed at www.netapp.com/TM are trademarks of NetApp, Inc.