IBM® Tivoli® Software

IBM Tivoli Workload Scheduler V9.1 Capacity Planning Guide

Document version 1.0

Pier Fortunato Bottan
Giorgio Corsetti
Claudio Fusi

Tivoli Workload Automation Performance Team - IBM Rome Lab

This edition applies to version 9, release 1 of Tivoli Workload Automation and to all subsequent releases and modifications until otherwise indicated in new editions.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Tivoli Workload Automation is a state-of-the-art production workload manager, designed to help you meet your present and future data processing challenges. Its scope encompasses your entire enterprise information system, including heterogeneous environments.
Pressures in today's data processing environment are making it increasingly difficult to maintain the same level of service to customers. Many installations find that their batch window is shrinking. More critical jobs must be finished before the morning online work 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.
Tivoli Workload Automation simplifies systems management across heterogeneous environments by integrating systems management functions. There are five main components in the portfolio:

1. Tivoli Workload Scheduler for z/OS
   The scheduler in z/OS® environments

2. Tivoli Workload Scheduler
   The scheduler in distributed environments

3. Tivoli Workload Scheduler for Applications
   It extends sophisticated workload automation to business enterprise resource planning (ERP) applications, such as SAP, PeopleSoft, and Oracle.

4. Tivoli Workload Scheduler Agent for z/OS
   With the Agent for z/OS you can define Tivoli Workload Scheduler jobs that run on the JES2 subsystem of z/OS.

5. Dynamic Workload Console (a web-based, graphical user interface for both Tivoli Workload Scheduler for z/OS and Tivoli Workload Scheduler).

Depending on the customer business needs or organizational structure, Tivoli Workload Automation distributed and z/OS components can be used in a mix of configurations to provide a completely distributed scheduling environment, a completely z/OS environment, or a “mixed” z/OS and distributed environment.

2. Scope

The performance objective of the tests described in this document, is to determine the Tivoli Workload Scheduler V9.1 scalability properties in the context of:

- New features (mirroring);
- Scheduling and user interface mixed workload;
- Scheduling on both fault tolerant and dynamic agents.

The ultimate target is to deliver capacity planning instruments to forecast which hardware and software configuration is needed to support a TWS workload. In this document some guidelines will be provided by mean of examples of configuration.

3. Capacity Test

3.1. Test Approach

In order to identify a rule to forecast hardware requirements for TWS capable to sustain a given workload, it was important to identify the scale properties of the software itself.

The combination of all possible configuration and workload is complex and large, for this reason from workload point of view, the following most significant variables have been chosen.

1. Number of jobs scheduled per unit of time;
2. Number of users concurrently working on the DWC console.

With regard to configuration optimization, capacity planning and performance tests that were performed for previous versions are used as a base for 9.1 with emphasis on the settings that were adopted to extend the capacity.

The general approach was to separate the user interface workload measurements from the scheduling ones and, once the linear superposition was verified (without interference factors), they could be merged together to allow resources usage forecast.

3.2. Test Benchmarks

3.2.1. User Interface Scenarios

A particular test scenario was chosen to provide backward comparison with the benchmark run in the previous release. Three main areas were identified and among them a set of subscenarios were designed with a defined weight as follows:

Monitoring (60% of the users)

1. All jobs in success (query result: 10K jobs) (15%)
2. All jobs in error (query result: 30 jobs) (15%)

3. All job streams in waiting (query result: 100 job streams) (15%)
4. All job streams in success (query result: 400 job streams) (15%)

Graphical (20% of the users)

5. Job stream view (25 jobs, 10 external deps, 28 internal dependencies) (10%)
6. Impact view (25 jobs, 2 external predecessors, 28 internal dependencies) (10%)

Modeling (20% of the users)

7. Job and job stream creation (10%)
8. Modify job (10%)

A Tivoli Workload Automation master with 100K jobs in plan was used. To keep constant the number of objects returned by monitoring queries, plan execution was kept blocked. Figures 5 - 12 explain each subscenario step. Each subscenario consists of three steps:

- Log in
- Transaction (composed of a series of activities that start from the primary dash welcome page and complete by returning to the same page)
- Log out

Each user in the automation framework (Rational Performance Tester) logs in and completes three transactions before logging out and reentering again with different credentials. The delay between each transaction is controlled by the framework to have a frequency of:

45 transactions/hour per user

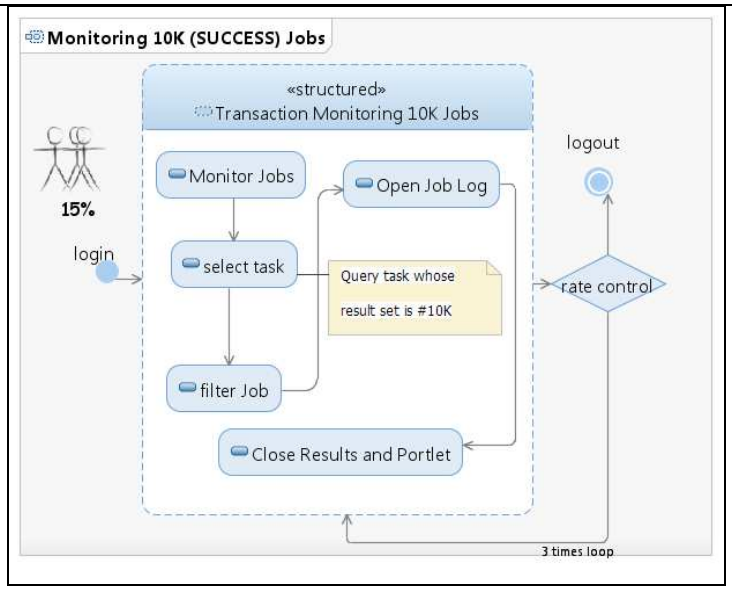

Figure 1. Monitoring jobs with "Success" status with 10^5 rows as result set.

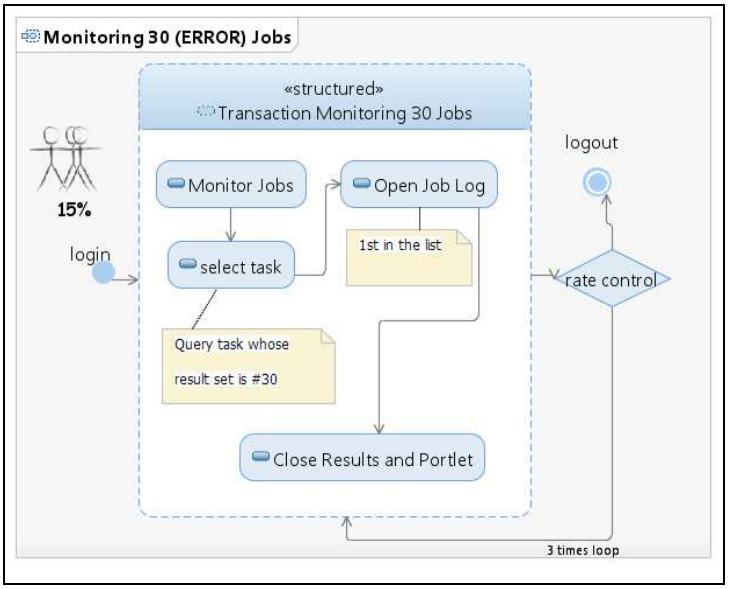

Figure 2. Monitoring jobs with "Success" status, obtaining 30 rows as result.

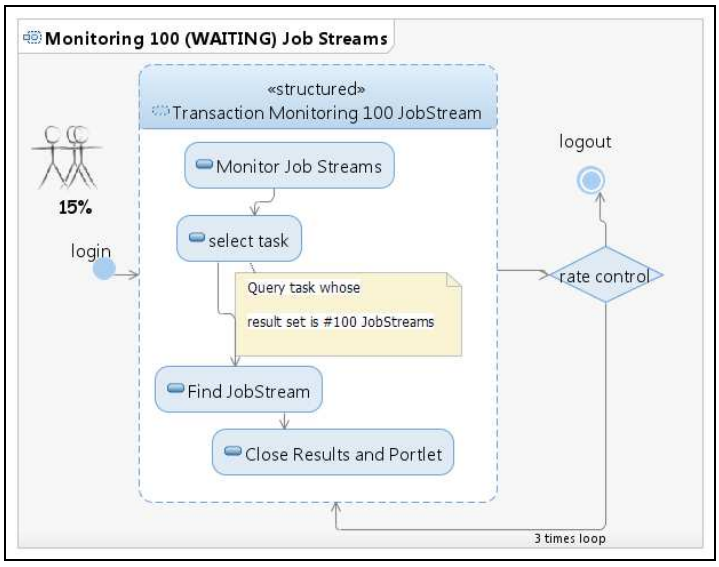

Figure 3. Monitoring job streams with “Waiting” status, obtaining 100 rows as result set.

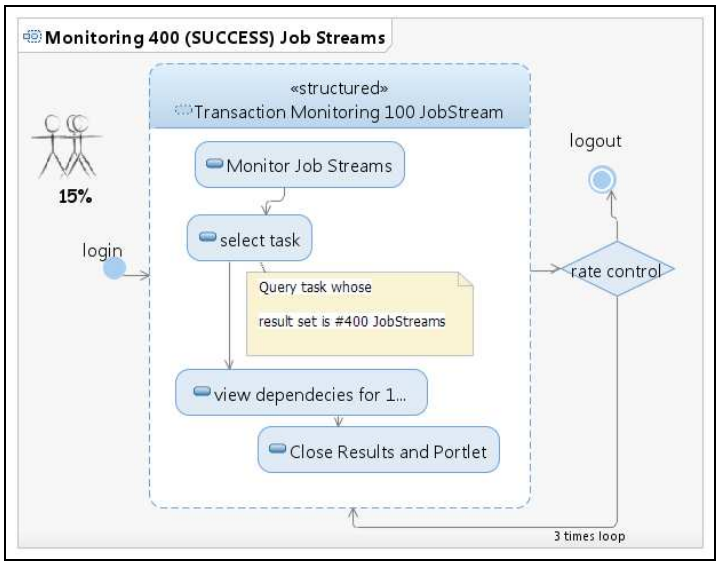

Figure 4. Monitoring job streams with “Success” status, obtaining 400 rows as result set.

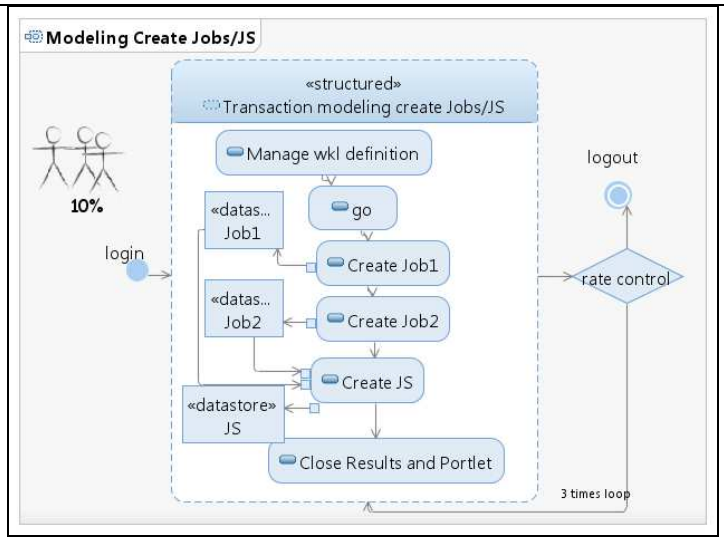

Figure 5. Creating jobs and job streams.

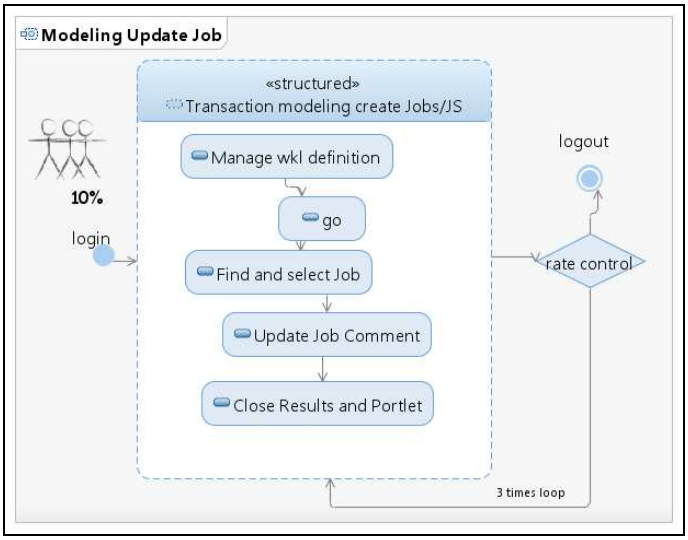

Figure 6. Updating jobs.

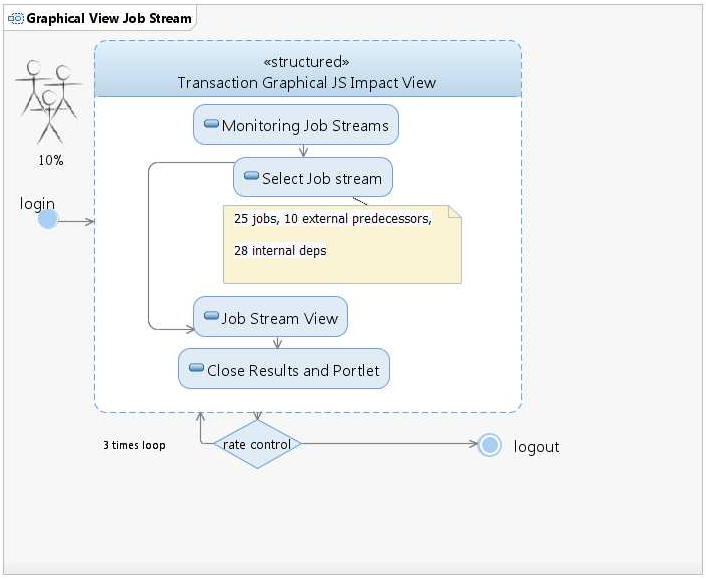

Figure 7. Graphical job stream view.

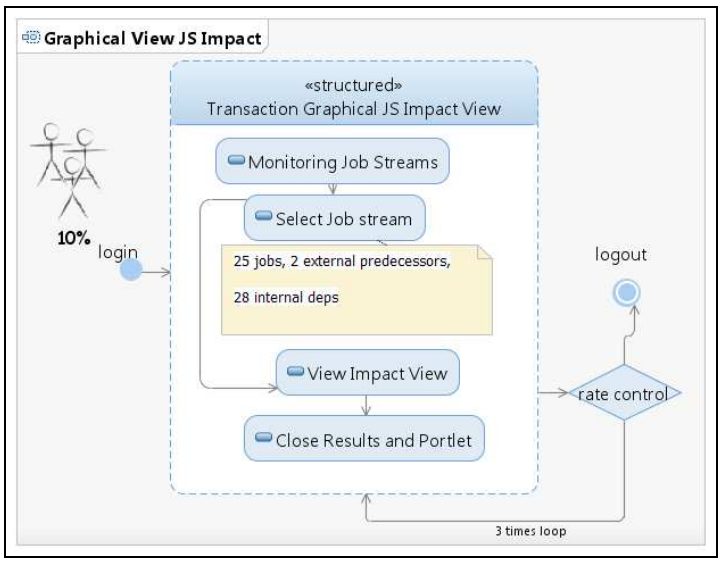

Figure 8. Graphical job stream impact view.

3.2.2. Scheduling Scenarios

The workload defined at master side was established to include these main guides: produce a constant background activity, also referred as baseline that runs for several hours and a peak activity that lasted 10 minutes. The ratio of Peak:Baseline has always been kept constant as 11 x factor.
Plan structure was defined as follows:

- 50 % Jobs executed on Fault Tolerant Agents
- 50% Jobs executed on Dynamic Agents

All Jobs were included in Job Stream composed by 50 Jobs and having the following structure: 25% of Job Streams with dependencies and 10 % of Jobs having dependencies from external Jobs (included in different Job Stream).

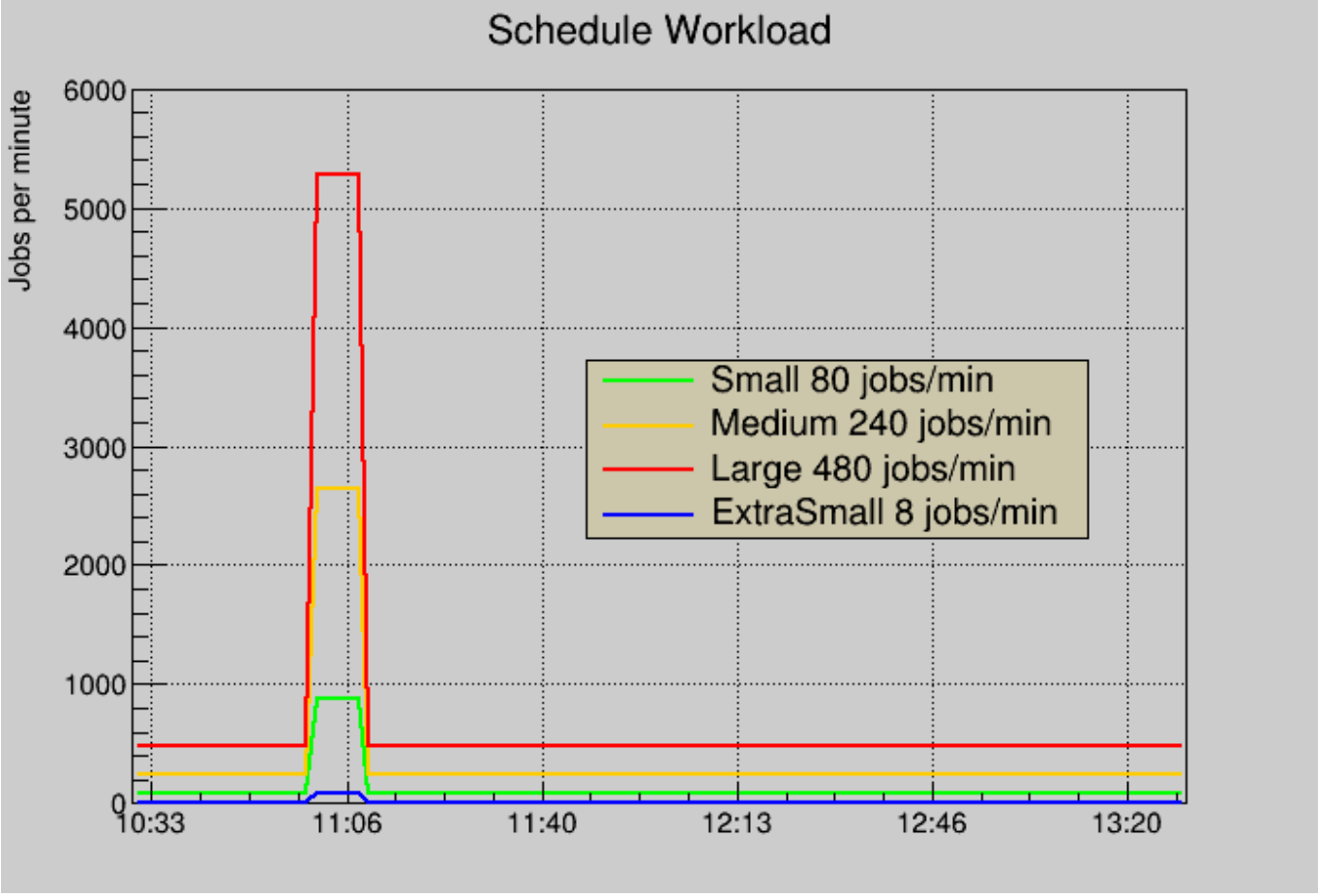

Figure 9. Scheduling workload benchmark design. Baselines and Peaks

Four schedule workloads have been designed characterized by number of jobs scheduled per minute.

	Xsmall (jobs/min)	Small (jobs/min)	Medium (jobs/min)	Large (jobs/min)
baseline	8	80	240	480
Peak (10 min)	88	880	2660	5280

Table 1. Schedule workload.

3.2.3. Environment

The test environment was based on LPAR nodes hosted on a P7 IBM 8233-E8B (3 GHz). All tests were performed in a 10 GB local area network. LPAR had dedicated cores whose numbers have been changed during benchmarks execution.

The following table summarizes the software used and its version

OS	AIX 7.1
RDBM	IBM DB2 v10.1.0.0
J2EE	IBM WebSphere Application Server 8.5.0.1 (JDK 7.0.2.0)
WebServer - Load Balancer	IBM HTTP Server 7.0.0.17+ WebSphere http Plugin
LDAP	IBM Directory Server 6.3
TWS 9.1	

Table 2. Software level of code

The https protocol was used and an IBM Http Server with IHS WebSphere Application Server Plugin acted as load balancer with “Random” policy to distribute user load on Tivoli Dynamic Workload Console servers. The procedure described at the following link:

http://pic.dhe.ibm.com/infocenter/tivihelp/v47r1/index.jsp?topic=%2Fcom.ibm.tivoli.itws.doc_9.1%2Fdistr%2Fsrc_tsweb%2FGeneral_Help%2FManagingSettingsRepository.htm

was followed to set up a high availability configuration (also known here as cluster).

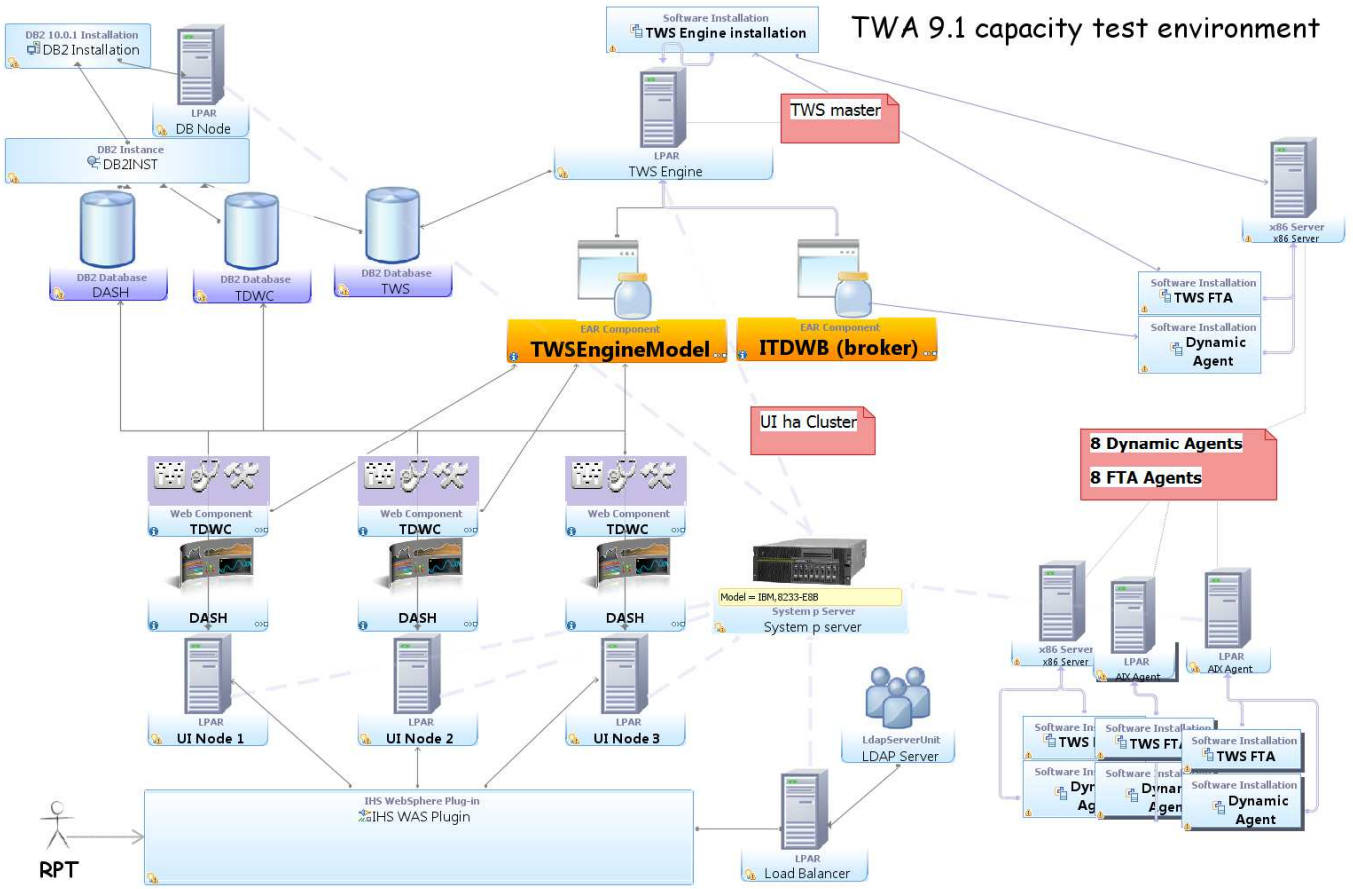

Figure 10. Overall view of Tivoli Dynamic Workload Console cluster environment

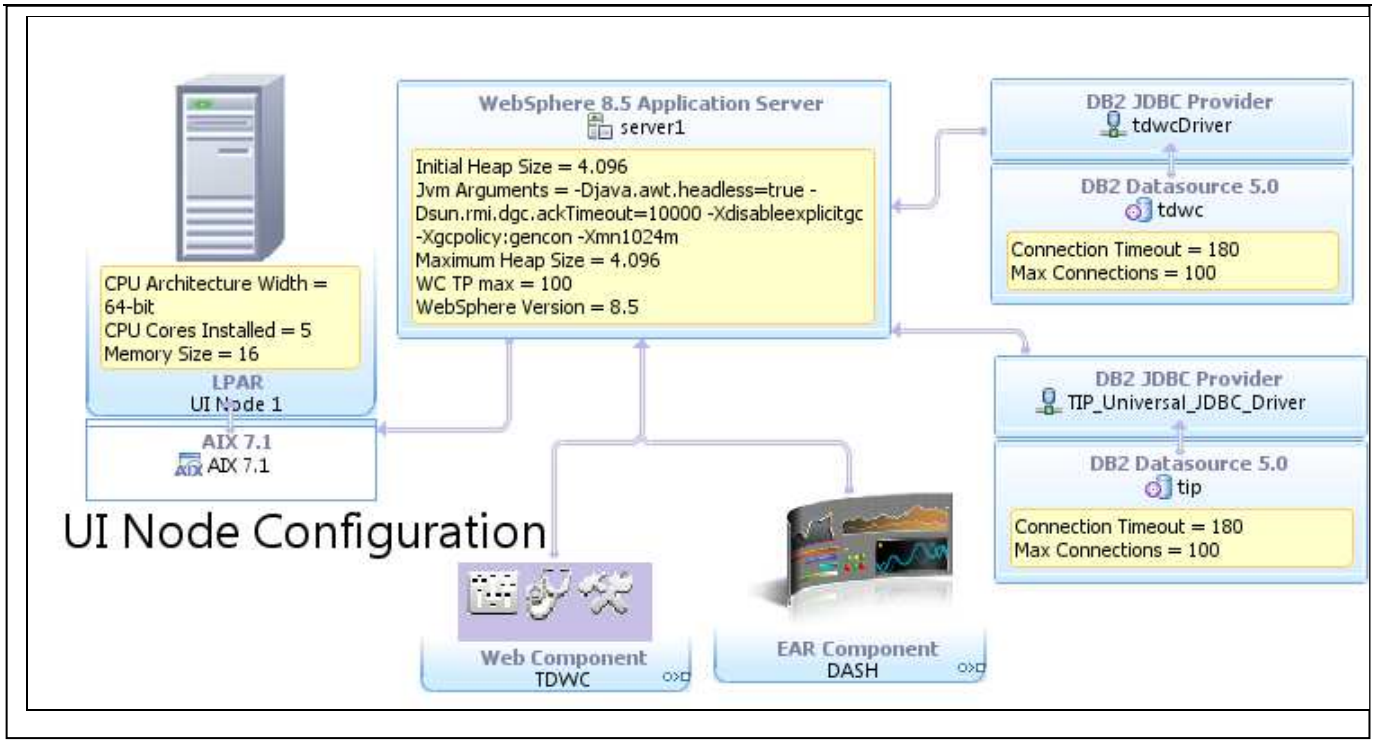

Figure 11. Tivoli Dynamic Workload Console node configuration

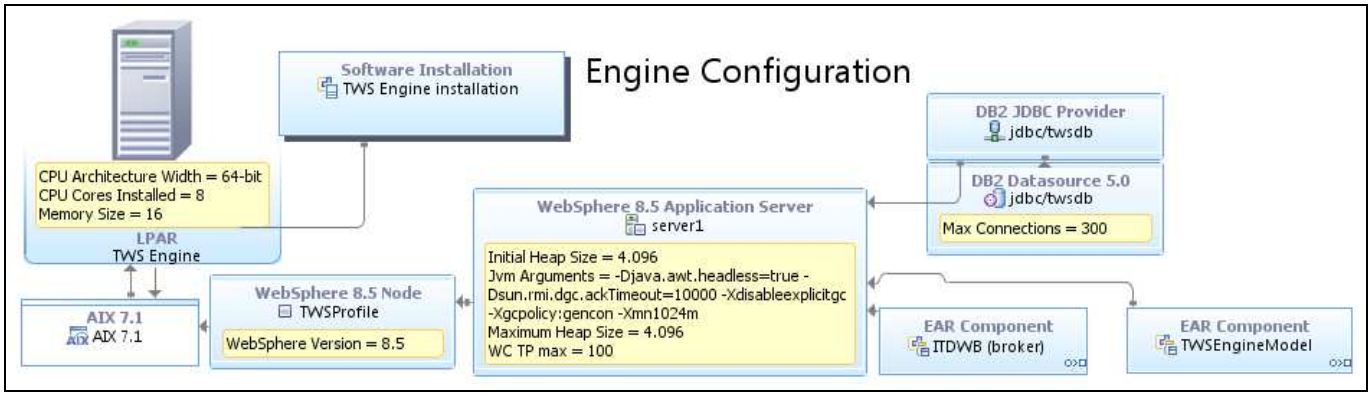

Figure 12. Engine node configuration

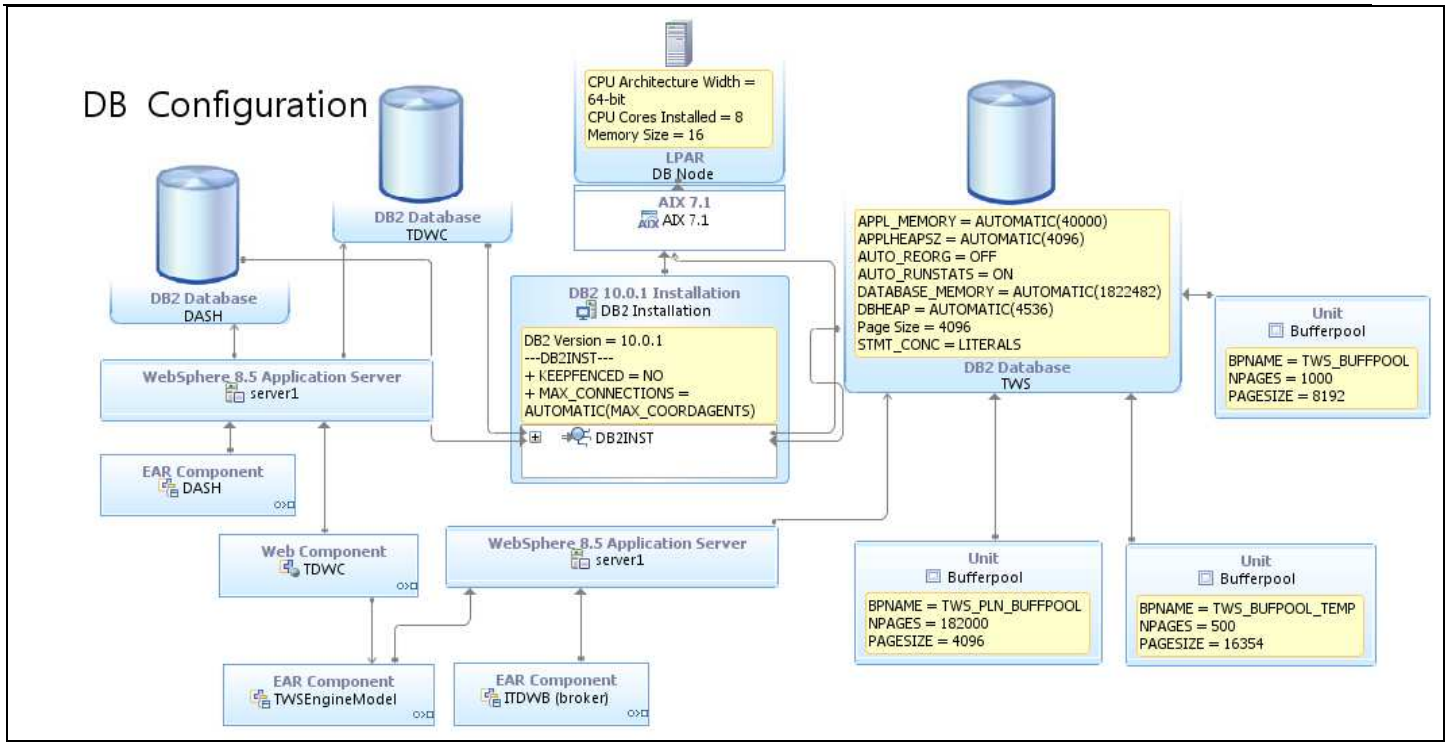

Figure 13. DB node configuration

3.2.4. Scenarios List

The following table summarizes the scenarios that have been executed

			Comments	Core assigned to each node	Agent connected
DWC UI scenarios	A	100-150-200 users	1 node	5	
	B	150-225-300 users	1 node	5	
	C	300-350-400 users	1 node	5	
	D	150-225-300 users	DWC Heap size 6 GB test	5	
	E	200-400-600 users	3 UI nodes in high availability configuration	5	
Scheduling scenarios	F	Extra small		1	1 agent
	G	Small	Scheduling test including 50 user UI activity	4	4 agents
	H	Medium		4	12 agents
	I	Large		4	16 agents

Table 3. Summary table of sampling benchmark.

3.3. Test tools

Rational Performance Tester (RPT) 8.3 was used to generate traffic and run a multiple users scenario. RPT also provides a response time for each http action on the browser by reporting the time spent on the server to process the request. RPT cannot determine the time spent by the browser to process data to be interpreted; standard monitoring tools and methodologies were used, such as nmon and IBM Support Assistant 4.0.1 – Garbage Collection and Memory Visualizer.

The Perfanalyst tool was used to control the middleware configuration and to analyse the DB2 snapshot

(https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=28cb6d68-ab67-4203-96f9-5538e654a5ff).

4. Benchmark Results

4.1. User Interface Scenarios

4.1.1. Single Node

User Interface scenarios tests (A-B-C-D) have been executed accordingly to the design described in section 3.2.1. Results allow to identify the scalability law trend for a DWC single node component. Assuming the number of concurrent user as an indicator to describe the incoming UI throughput and the ratio users/response-time as a good way to characterize the DWC capacity to serve, the scalability behavior is described in Figure 14. As the curve evidences, the best number of users per DWC node (assuming 4 GB heap size) is within the 200-250 users range.

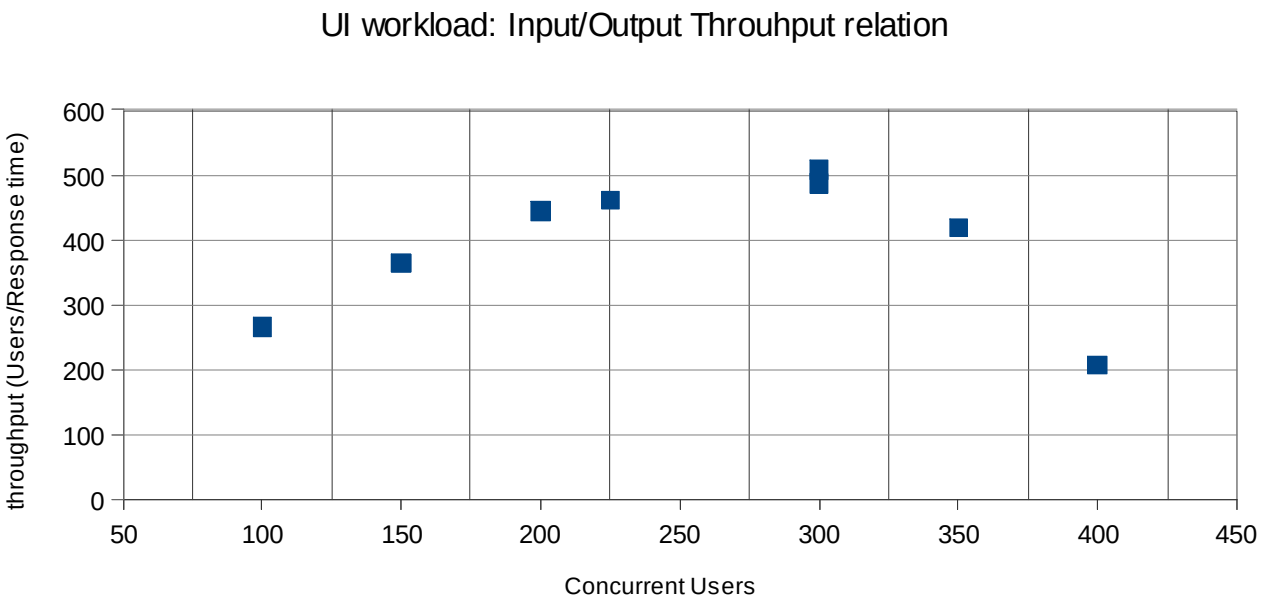

Figure 14. Incoming throughput, in terms of concurrent users, vs output throughput, in terms of service time to process a request.

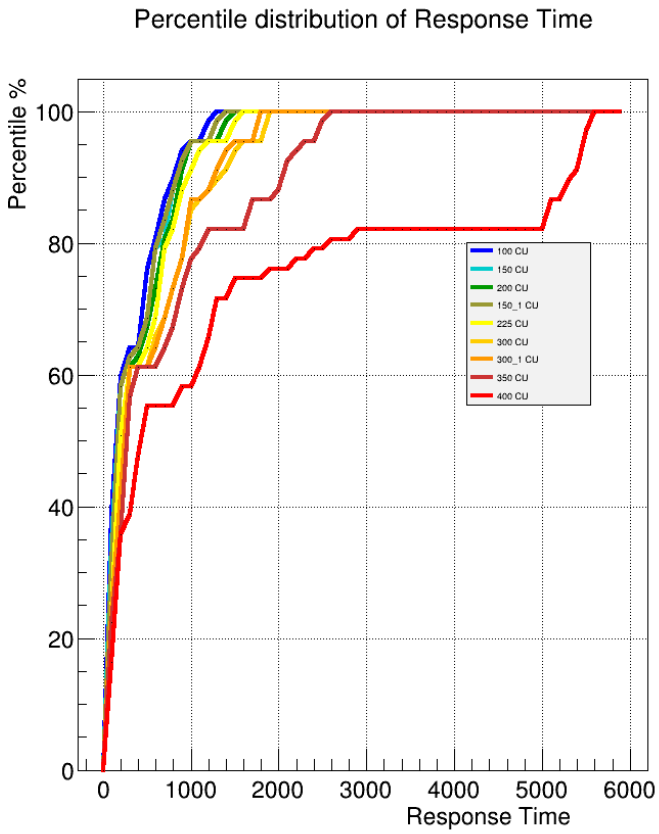

Figure 15. Percentile distribution curve of response time taken at different concurrent users workloads.

Garbage collector analysis emphasizes how total pause time could one of the most important limiting factor in addition to database load and user interface node CPU load.

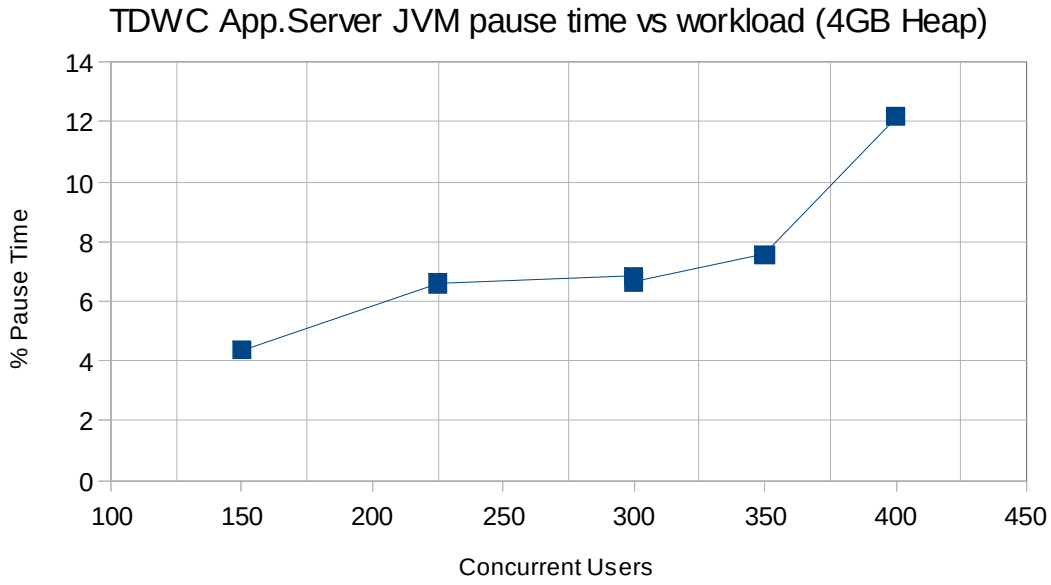

Figure 16. Total pause time percentage.

Single Node UI Scenario with increase Heap Size (D) has been executed to understand the performance impact of a greater heap size. A 6 GB heap size configuration has been applied. Comparing the 300 users workload these are the main results:

- 16% response time improvements;
- 6% CPU overload.

4.1.2. Multiple Nodes

TWA 9.1high concurrency DWC scenarios results have been already published in Tivoli Workload Automation 9.1 DWC whitepaper document. In that context a 5 nodes high availability configuration has been used to support 500 concurrent users (100 users per node). Leveraging the results reported in section 4.1.1, the same total workload has been tested with 3 nodes (200 users per node).

4.2. Scheduling Scenarios

Scheduling scenarios (F,G,H,I) have been executed accordingly to configuration described in sec. 3.2.2.

Differently form Fault Tolerant Agents, dynamic agents, managed by Dynamic Domain Master jobs submission is handled centrally and causes additional processing at Master node. The workflow of a dynamic job submission is described in Figure 17 ; the experimental result of the benchmarks executed in this context showed some critical steps in the workflow. Specifically the most armful ring in the chain is the mailman server queue processing that feeds the Broker component. For this reason it has been chosen to add an additional Mailman Server that

specifically serves the Broker component.

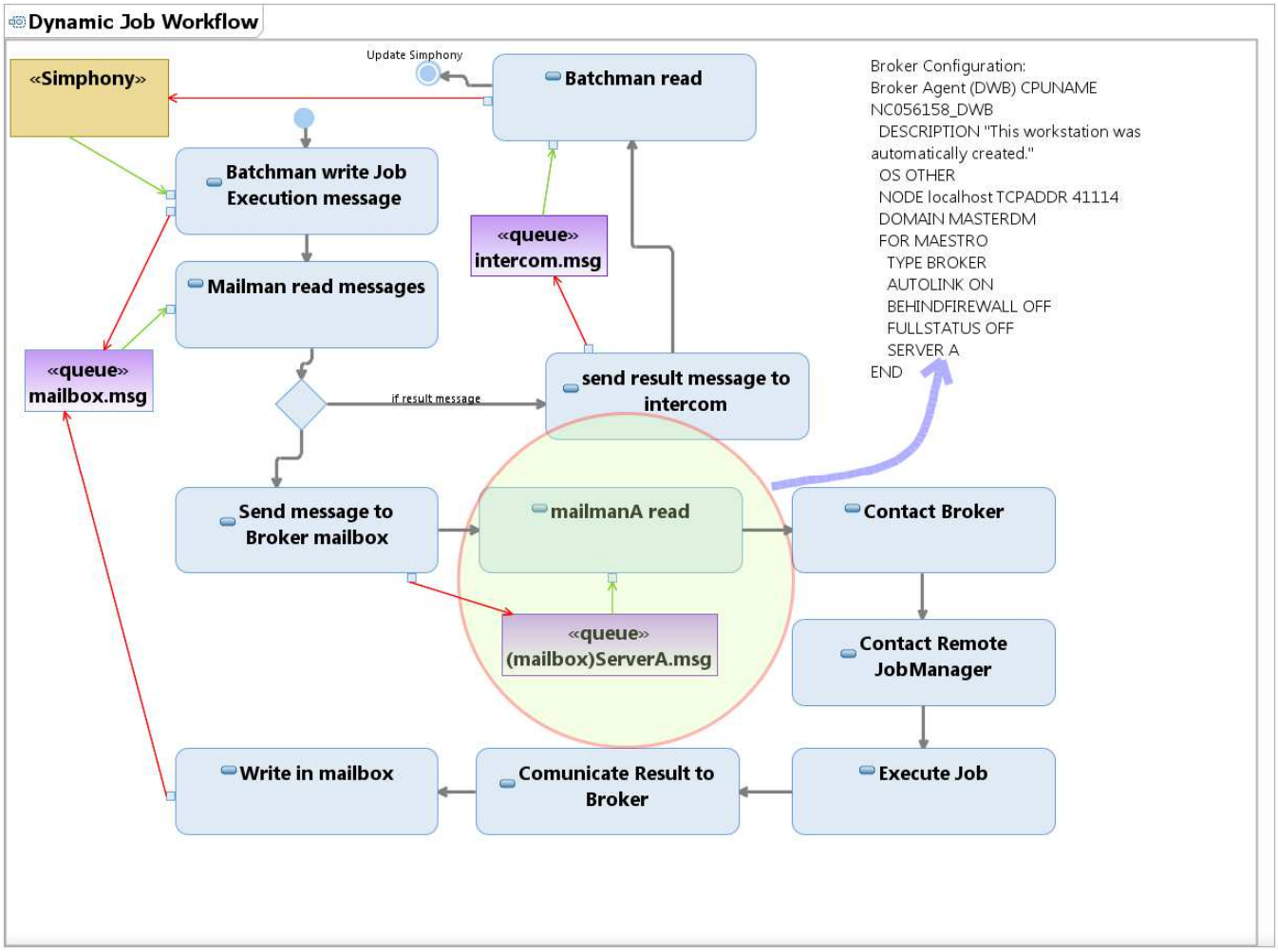

Figure 17. Dynamic Agents job submission workflow. The Dynamic Domain Manager (Broker) queue processing activity has a limited throughput. In order to separate this activity from the other Mailman Server ones, a specific Mailman Server process, and queue, has been defined using the Dynamic Agent Broker configuration. In this example the new server is call Server A.

4.2.1. Scaling Properties

The output throughput of Dynamic Jobs submission reach up to 400 jobs submitted per minute (Figure 18). The effect of this behavior is the growing of mailman server queue and a possible delay in job submission.
For this reason it could be advisable to add an additional Dynamic Domain Manager to help in balancing dynamic agents workload.

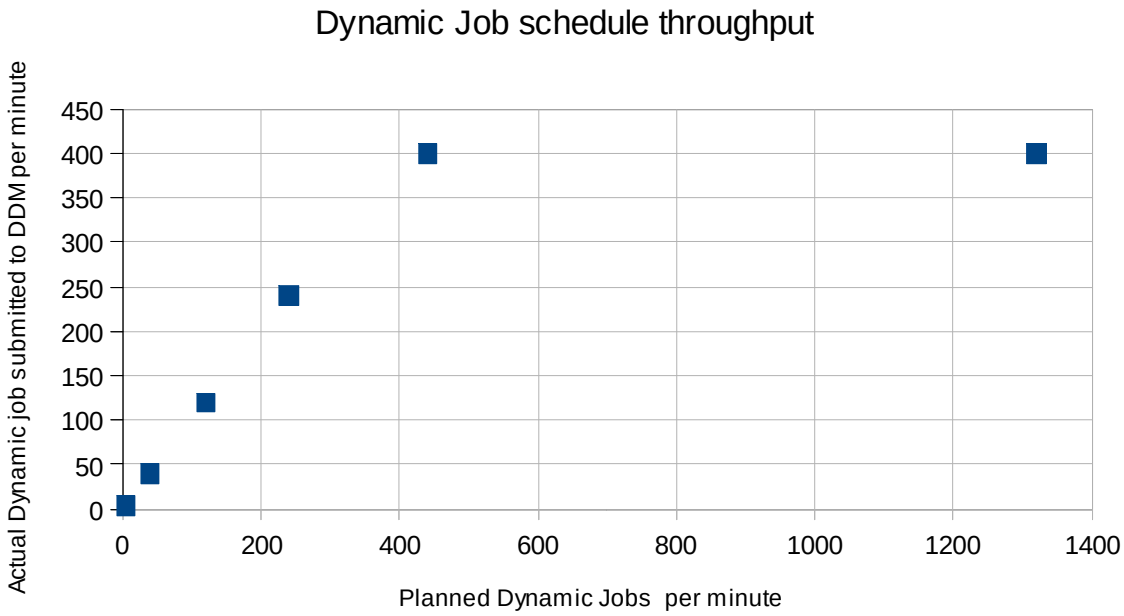

Figure 18. Incoming throughput, in terms of planned scheduled jobs on Dynamic Agents, vs output throughput, in terms of actual scheduled jobs.

All scheduling reports are uploaded and enqueued into mirroring queue ready to be inserted in the database. Mirroring queue processing reaches a throughput able to support up to 300 jobs submission per minute (causing up to 900 events per minute); if input throughput is grater than this number, mirroring queue size starts to increase until the maximum size is reached (10 MB was the limit in this context) and a full plan resynch to database is triggered.

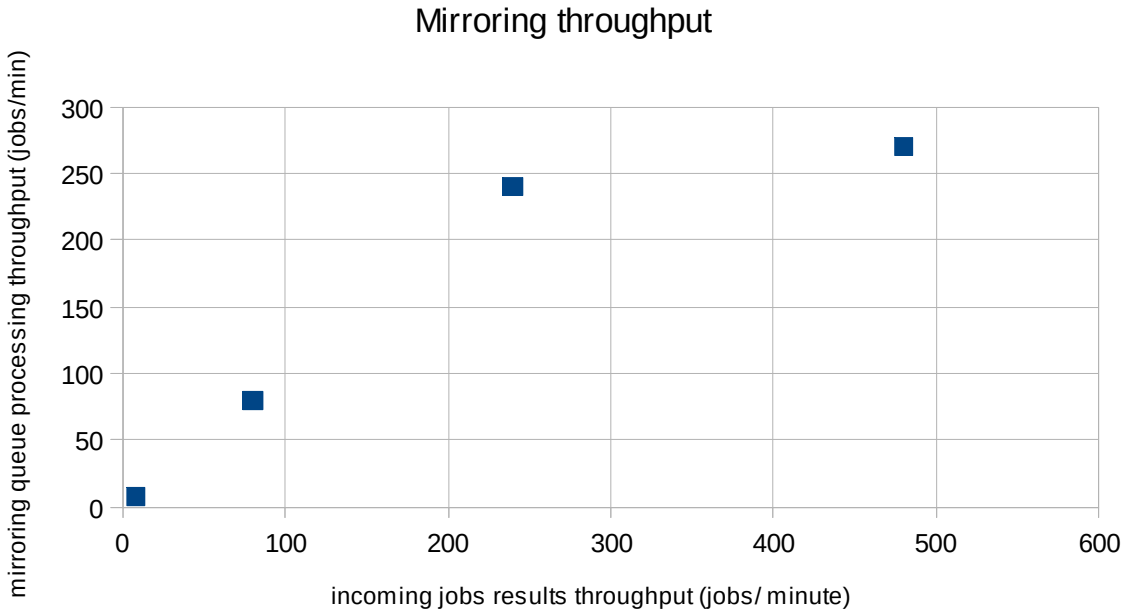

Figure 19. Incoming throughput, in terms of jobs results, vs output throughput, in terms of mirroring queue events processed per unit of time.

4.2.2. CPU Usage Fault Tolerant Agent vs Dynamic Agent Job Submission

As it has been illustrated in section 4.2 , Dynamic Agent job submission flow is different from Fault Tolerant Agent one. The executed scenarios have a balanced mixed workload of both types. Additional test put in evidence the different contributes coming from different submission type. Since to the maximal throughput is reached the slope for dynamic submission is remarkable.

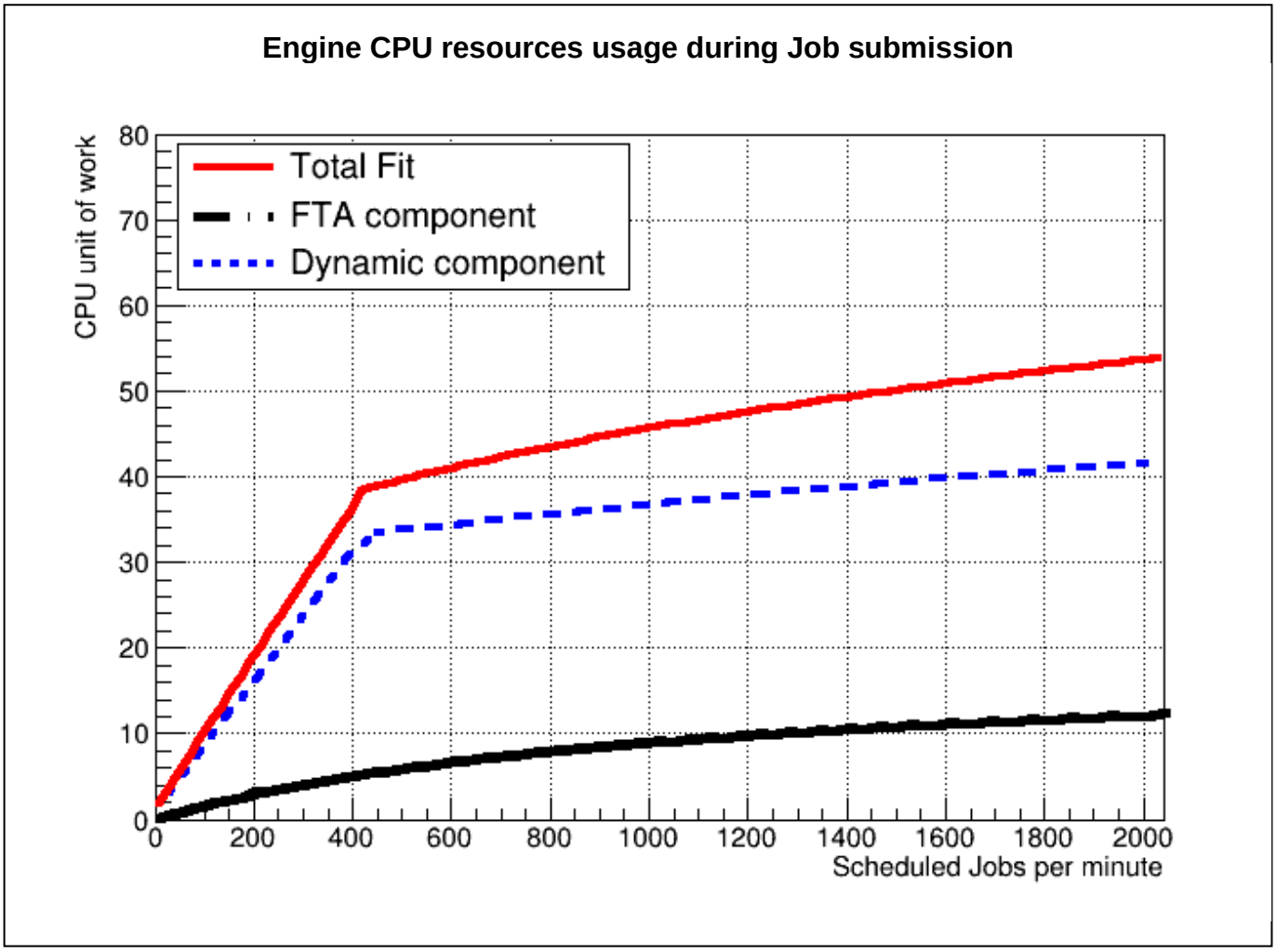

Figure 20. Different CPU usage contributes coming from Dynamic(50%) and Fault Tolerant(50%) agents job submission.

5. Recommendations

5.1. CPU Capacity

All tests described in this document have been executed on **P7 IBM 8233-E8B 3GHz** processors assigned exclusively to LPAR (no shared pools or capping feature have been applied). While planning correct CPU sizing, the information provided in Table 7 could be a reference point to start. It has been demonstrated the validity of superposition property that makes possible to assume that the resource usage could be considered as the sum of UI (DWC) part and Engine one.

5.2. Memory

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

Concurrent users range	1 – 50	50 -100	100 -200
DWC heap size	1 GB	2 GB	4 GB

Table 4. Engine Websphere Application Server heap configuration

Schedule (jobs per min)	1 – 50	50 -100	>200
TWS Engine heap size	1 GB	1.5 GB	2 GB

Table 5. DWC Websphere Application Server heap configuration

In addition to the above memory requirements the native memory for java process and TWA process should be taken into consideration.

5.3. Topology

All scenarios have been executed in a single Master, controlling Domain Manager / Dynamic Domain Manager, topology, to evaluate the scalability behavior of the basic configuration.

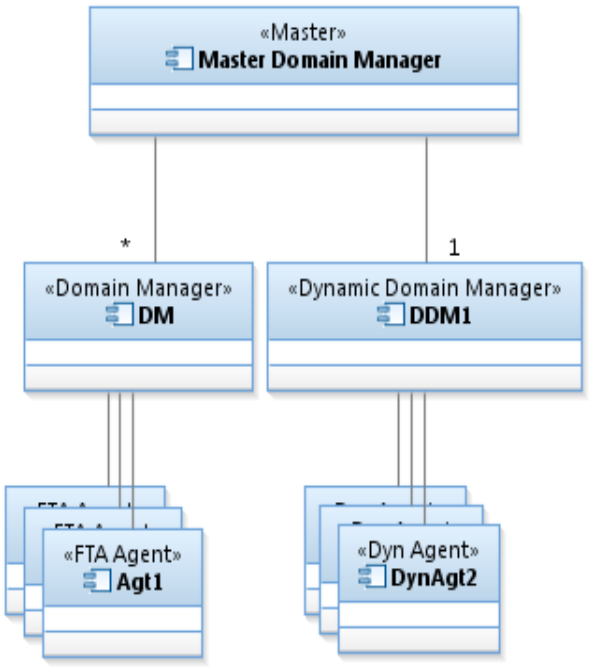

Figure 21. Single Dynamic Domain Manager Topology

In section 4.2.1 it has been seen the behavior of dynamic job submission throughput. The overall throughput could be increased adding an additional Dynamic Domain Manager component to the actual topology that helps in managing dynamic agents.

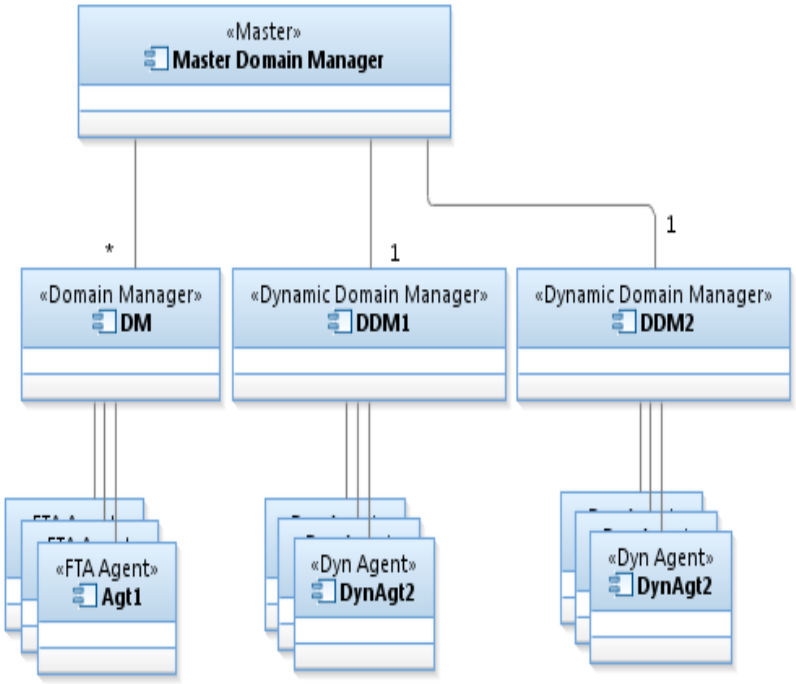

Figure 22. Single Dynamic Domain Manager Topology.

5.4. Tunings and Settings

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

	Parameter	Value	Comment
UI Node	Tivoli Dynamic Workload Console configuration settings repository (see http://pic.dhe.ibm.com/infocenter/tivihelp/v47r1/topic/com.ibm.tivoli.itws.doc_9.1/distr/src_tsweb/General_Help/Changing_settings_repository.htm)	Use database as settings repository	It is strongly recommended to adopt this configuration to allow acceptable UI performances
	WebSphere Application Server WC Thread Pool Size	100	Should be adjusted with number of concurrent users accordingly
	WebSphere Application Server JVM max heap = min heap	4096 for 200 users and could be reduced according to the number of users	
	WebSphere Application Server JVM options	-Djava.awt.headless=true -Dsun.rmi.dgc.ackTimeout=10000 -Xdisableexplicitgc -Xgcpolicy:gencon –Xmn1024m	-Xmn parameter value should be ¼ of total heap size
	WebSphere Application Server JDBC max Connections	100	
Tivoli Workload Scheduler engine	WebSphere Application Server JDBC max Connections	300	
	WebSphere Application Server JVM max heap = min heap	2048	
	WebSphere Application Server JVM options	-Djava.awt.headless=true -Dsun.rmi.dgc.ackTimeout=10000 -Xdisableexplicitgc -Xgcpolicy:gencon –Xmn 512m	
DB	Dbm KEEPFENCED	NO	
	dbm dbMAX_CONNECTION	AUTOMATIC	
	Db STMT_CONC	LITERALS	This setting optimizes query executions and reduces

				CPU usage
	Db APPL_MEMORY, APPLHEAPSZ, DATABASE_MEMORY, DBHEAP		AUTOMATIC	
	Db AUTO_RUNSTAT		ON	
	AUTO_REORG		OFF	
	TWS_PLN_BUFFPOOL	NPAGES	182000	
		PAGESIZE	4096	
	TWS_BUFFPOOL_TEMP	NPAGES	500	
		PAGESIZE	16354	
	TWS_BUFFPOOL	NPAGES	8192	
		PAGESIZE	1000	

TDWB	JobDispatcherConfig.properties	Queue settings	`Queue.actions.0 = cancel, cancelAllocation, cancelOrphanAllocation`
			`Queue.size.0 = 10`
			`Queue.actions.1 = reallocateAllocation`
			`Queue.size.1 = 10`
			`Queue.actions.2 = updateFailed`
			`Queue.size.2 = 10`
			`Queue.actions.3 = completed`
			`Queue.size.3 = 30`
			`Queue.actions.4 = execute`
			`Queue.size.4 = 30`
			`Queue.actions.5 = submitted`
			`Queue.size.5 = 30`
			`Queue.actions.6 = notification`
			`Queue.size.6 = 30`	

	ResourceAdvisorConfig.properties	MaxAllocsPerTimeSlot	`1000`	
		TimeSlotLength	`10`	
		MaxAllocsInCache	`50000`	

Table 6. Main configurations and tunings.

6. Capacity Plan Examples

In this work the number key parameters used to identify the workload has been kept as simple as possible:

1. Number of concurrent users assuming a mixed scenarios like the one described in section 3.2.1;
2. Number of jobs to be scheduled;
3. Percentage of dynamic Jobs to schedule.

With the above inputs it is possible to forecast the resources needed to host TWA 9.1. Internal fit functions have been used to model the workload and resource usage relationship. It has been considered 65 % CPU usage as threshold to request additional core.

In this section some examples of capacity planning are reported. It must be remembered that all the requirements are related to **PowerPC P7** platform; nevertheless this information could be used as reference point for different platform architecture.

	NODE	Cores capacity	Disk throughput Read-Write (MB/sec)	Network throughput Read-Write (MB/sec)	RAM capacity (GB)
250K jobs (50% FTA +50% DYN) per day (175 jobs/min) 100 concurrent users					
3Nodes	**TWS-Engine**	2	0-0.5	1-1	3
	RDBMS	1	2-0.5	0.5-1.5	5
	DWC	2	0-0.1	1.2-1	6
500K jobs (50% FTA +50% DYN) per day (350 jobs/min) 100 concurrent users					
3Nodes	**TWS-Engine**	2	0-1	0.9-2	4
	RDBMS	2	2.3-0.9	0.5-1.5	5
	DWC	2	0-0.1	1.2-1	6
750K jobs (50% FTA +50% DYN) per day (485 jobs/min) 100 concurrent users					
3Nodes	**TWS-Engine**	3	0-1.3	1.6-1.3	4
	RDBMS	3	2.3-1.2	1-2.2	5
	DWC	2	0-0.1	1.2-1	6
10K jobs (50% FTA +50% DYN) per day (8 jobs/min) 20 concurrent users					
1Node	**TWS-Engine RDBMS DWC**	1	0.5-0.1	0.5-0.7	5

Table 7. Capacity planning samples.

The above capacity planning examples are referred to the workload described in section 3.2 . In particular, they are based of 50% job dynamic agent job scheduling. If the ratio changes, TWS-Engine Cpu capacity requirement changes. For example, assuming that all agents are dynamic (100%) the following configuration should be considered:

	NODE	Cores capacity	Disk throughput Read-Write (MB/sec)	Network throughput Read-Write (MB/sec)	RAM capacity (GB)
500K jobs (100% DYN) per day (350 jobs/min) 100 concurrent users					
3Node	**TWS-Engine**	**3**	0-1	0.9-2	4
	RDBMS	2	2.3-0.9	0.5-1.5	5
	DWC	2	0-0.1	1.2-1	6

Table 8. Workload impact of 100% dynamic agent job scheduling.

7. Notices

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

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

IBM 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:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION 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. IBM 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-IBM 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 IBM product and use of those Web sites is at your own risk.

IBM 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:

IBM Corporation
2Z4A/101
11400 Burnet Road

Austin, TX 78758 U.S.A.

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 IBM under terms of the IBM Customer Agreement, IBM International Program 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-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

All IBM prices shown are IBM'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

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

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

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

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

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