IBM® Tivoli® Software

IBM Tivoli Workload Automation V9.1 Distributed Dynamic Workload Console Performance in High Concurrency Context

Document version 1.0

Pier Fortunato Bottan
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 distributed-only scheduling environment, a z/OS-only environment, or a “mixed” z/OS and distributed environment.

2. Scope

A high number of concurrent users performing monitoring activities from the Dynamic Workload Console can result in slow response times and overall performance. Monitoring activities require access to the TWS plan included in the Symphony file of the TWS master domain manager. With Tivoli Workload Automation V9.1, significant rework was done to remove the limiting factors for user interface (UI) responsiveness, especially in the high concurrency context. By replicating the plan data in a database, the data can be accessed quickly and reliably.

The objective of this document is to demonstrate how Tivoli Workload Automation V9.1 can support 500 concurrent Dynamic Workload Console UI users more efficiently than in previous versions. A mixed workload of scheduling and UI activities was tested for an extended period to ensure that core capabilities were not impacted.

2.1. Executive summary

The most outstanding performance results compared to Tivoli Workload Automation V8.6FP1 are shown in the following table.

Dynamic Workload Console Area	Tivoli Workload Automation V9.1 vs V8.6FP1 improvements @ 500 concurrent users
Monitoring	Up to 21x
Modeling	Up to 15x
Graphical View	Up to 20x

Table 1. Tivoli Workload Automation V9.1 highest improvement factors with respect to V8.6

No drawbacks were detected during the 7-day run.

In general, from a capacity point of view, the CPU load was moved from the Tivoli Workload Automation engine to the database.

3. Environment configuration and tuning

The test environment was based on LPAR nodes hosted on a P7 IBM 8233-E8B. All tests were performed in a 10 GB local area network. LPAR had dedicated cores (5 for TDWC nodes, 8 for database and engine nodes with SMT=4)

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 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).

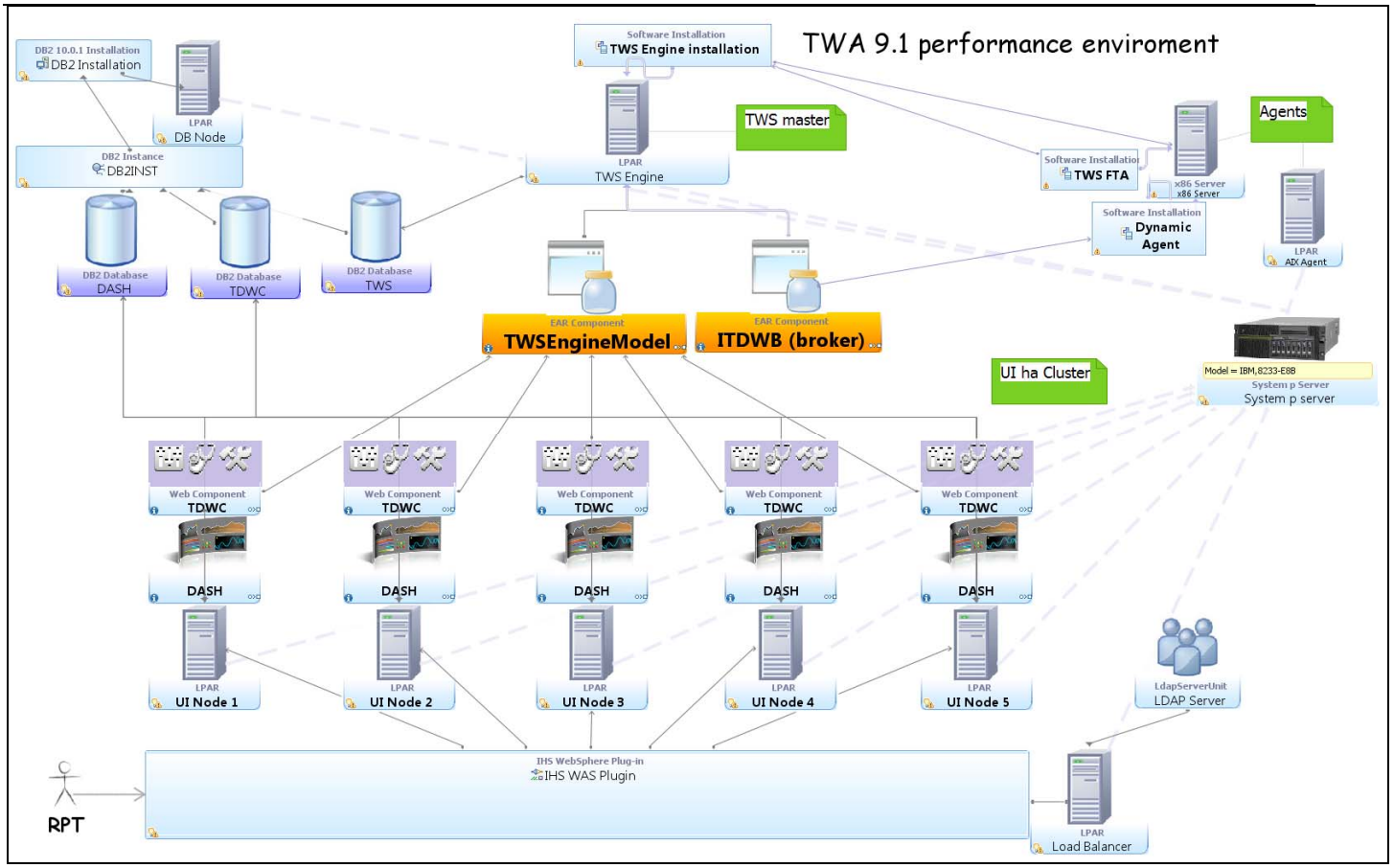

Figure 1. Overall view of Dynamic Workload Console cluster environment

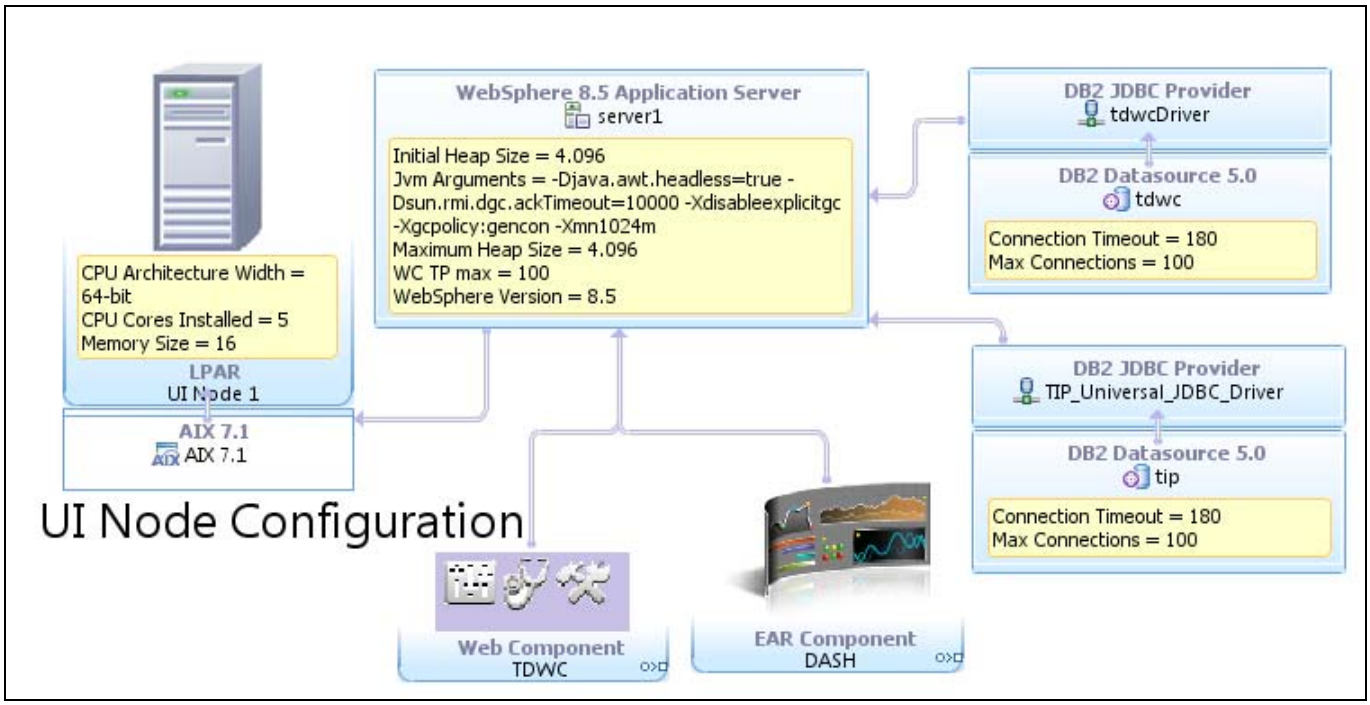

Figure 2. Dynamic Workload Console node configuration

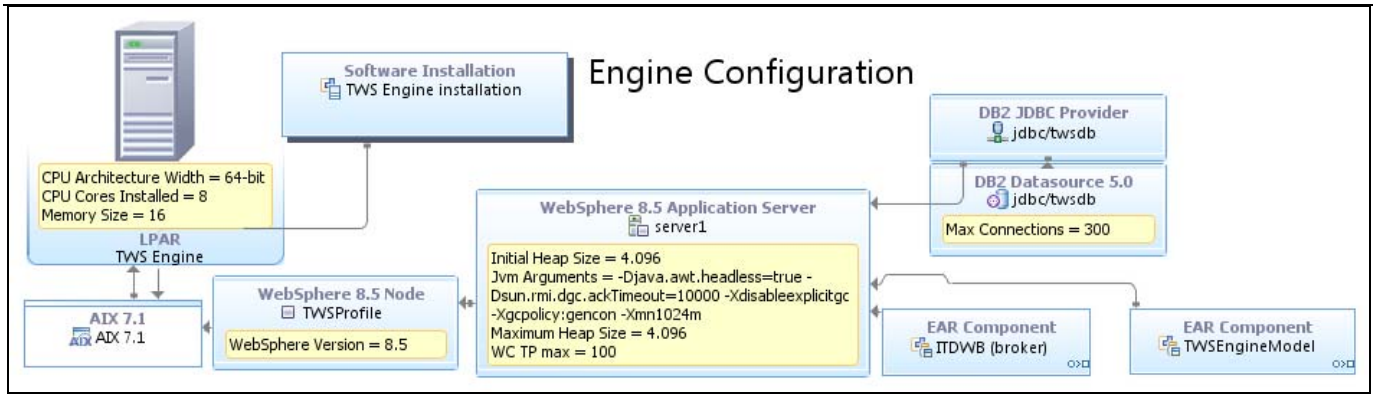

Figure 3. Engine node configuration

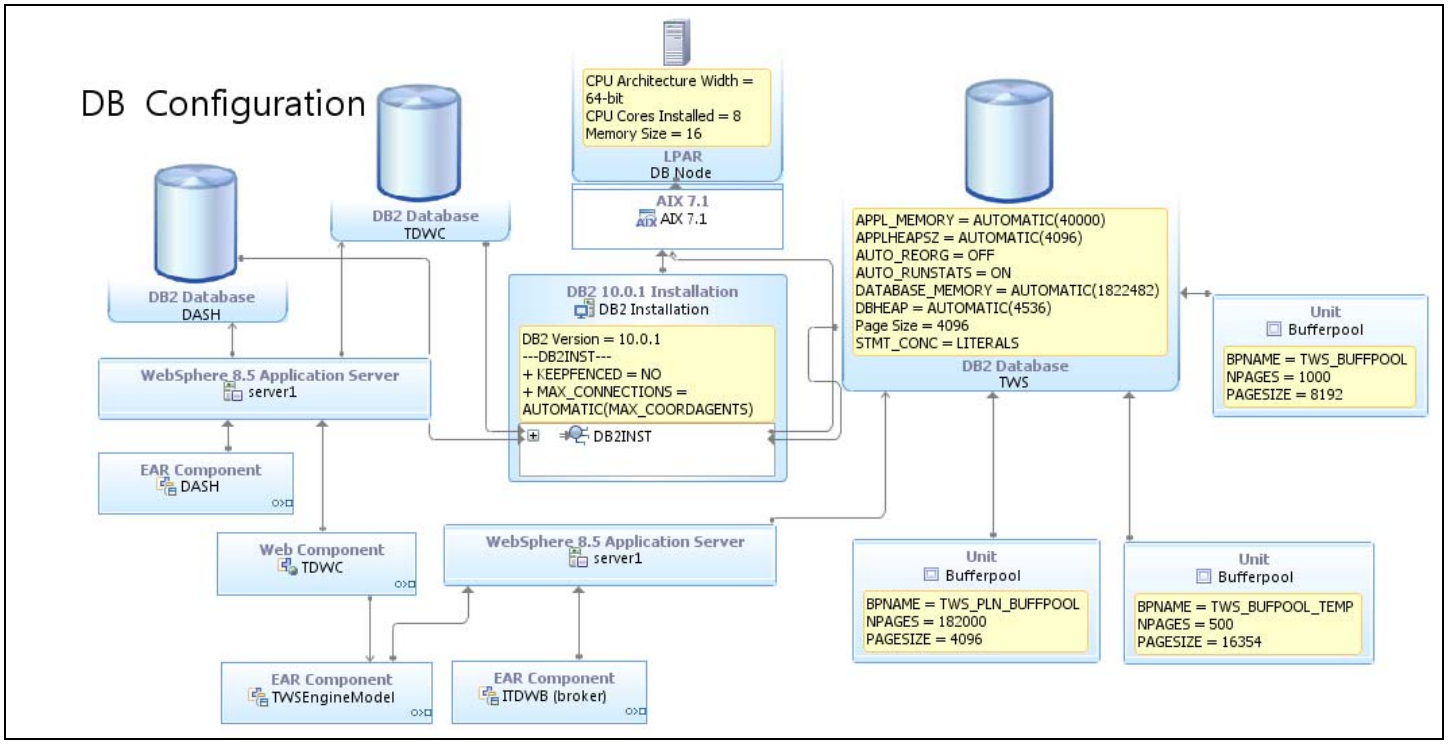

Figure 4. DB node configuration

3.1. Recommendations

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. Table 1 summarizes the most significant tunings.

	Parameter	Value	Comment
UI Node	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	
	WebSphere Application Server JVM options		-Djava.awt.headless=true -Dsun.rmi.dgc.ackTimeout=10000 -Xdisableexplicitgc -Xgcpolicy:gencon -Xmn1024m	
	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		4096	
	WebSphere Application Server JVM options		-Djava.awt.headless=true -Dsun.rmi.dgc.ackTimeout=10000 -Xdisableexplicitgc -Xgcpolicy:gencon -Xmn1024m	
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	

Table 2. Main configuration tunings

3.2. 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 against 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; a specific stopwatch tool was used to measure overall time against Firefox 17.0.6; 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. Dynamic Workload Console performance improvements

4.1. Scenarios

4.1.1. 500 concurrent users test

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 succ (query result: 10K jobs)
2. All jobs in err (query result: 30 jobs)
3. All job streams in waiting (query result: 100 js)
4. All job streams in succ (query result: 400 js)

Graphical (20% of the users)

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

Modeling (20% of the users)

7. Job and job stream creation
8. Modify job

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

- Log in
- Transaction (composed of a set of several activities that start from primary dash welcome page and complete by returning to it)
- Log out

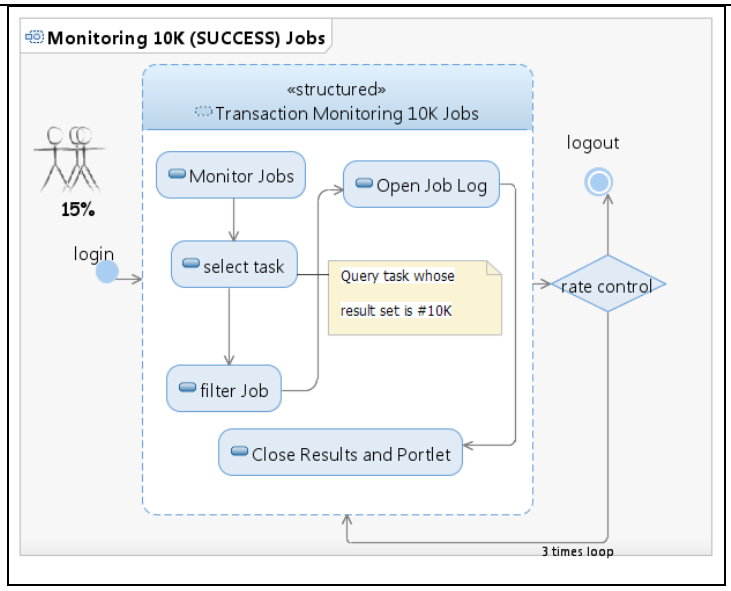

Figure 5. Monitoring jobs with "Success" status with 10^5 rows as result set

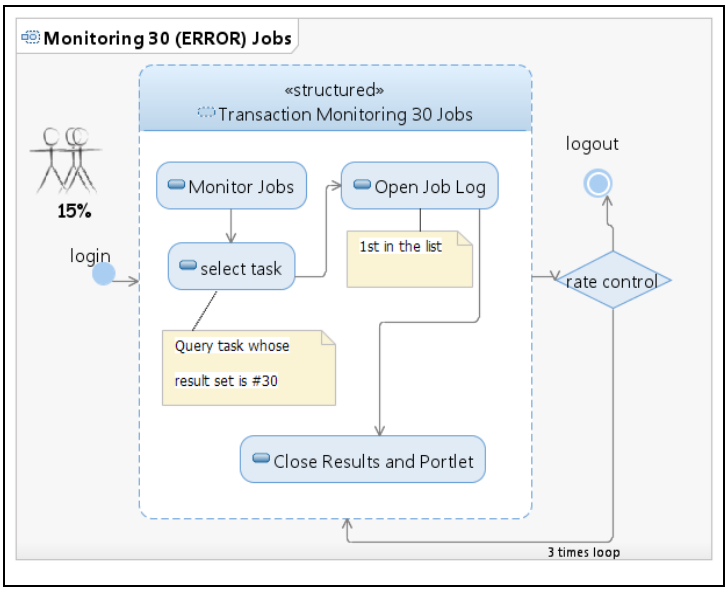

Figure 6. Monitoring jobs with "Success" status, obtaining 30 rows as result

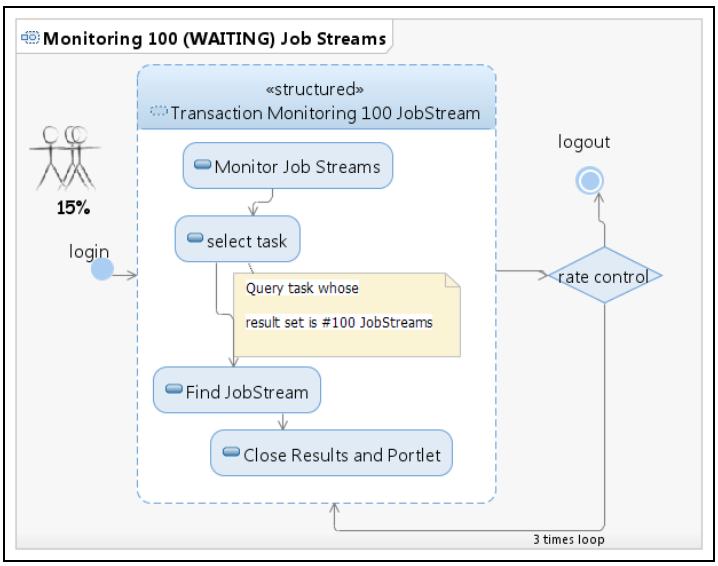

Figure 7. Monitoring job streams with “Waiting” status, obtaining 100 rows as result set

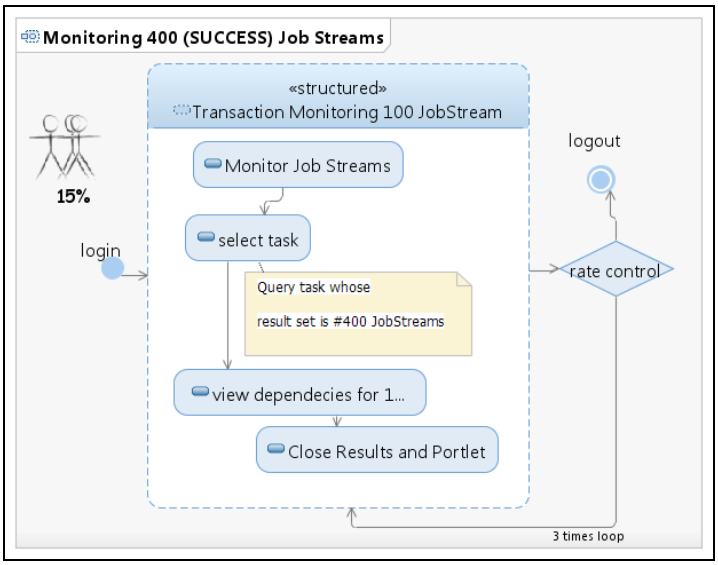

Figure 8. Monitoring job streams with “Success” status, obtaining 400 rows as result set

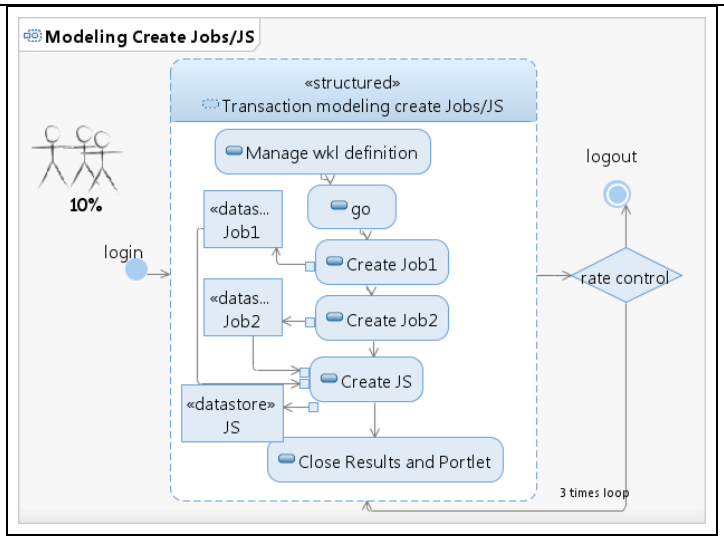

Figure 9. Creating jobs and job streams

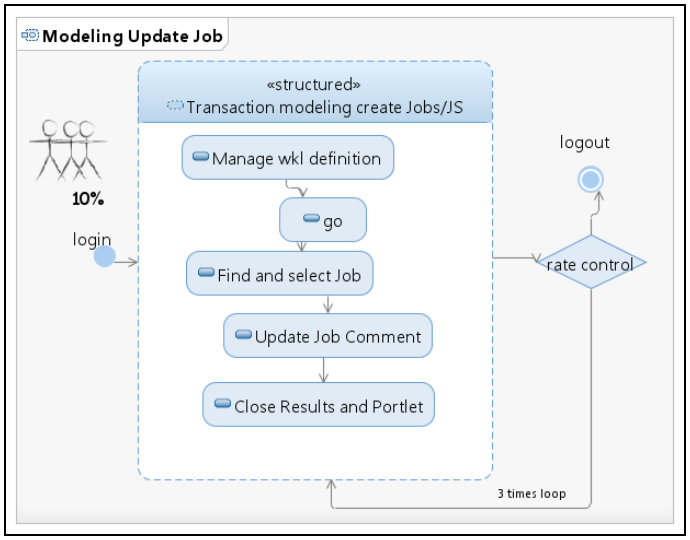

Figure 10. Updating jobs

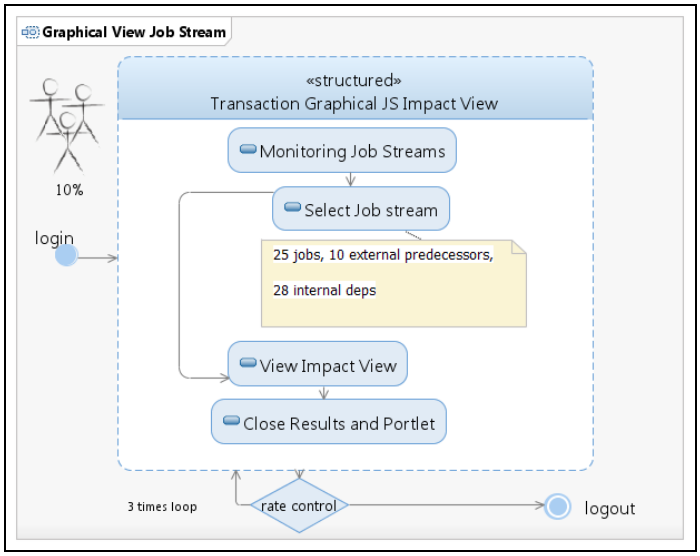

Figure 11. Graphical job stream view

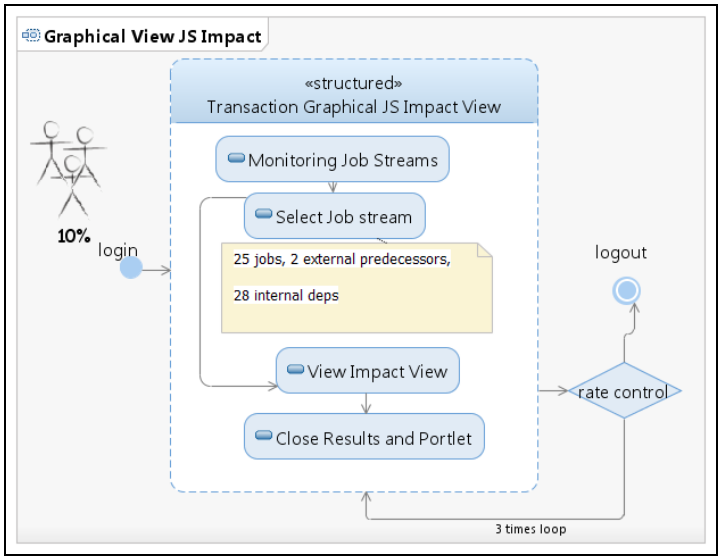

Figure 12. Graphical job stream impact view

4.1.1.1. Workload

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

Concurrency test was composed of three stages of 100, 300, and 500 users. The overall concurrency of 500 users is about:

50 https page requests per second

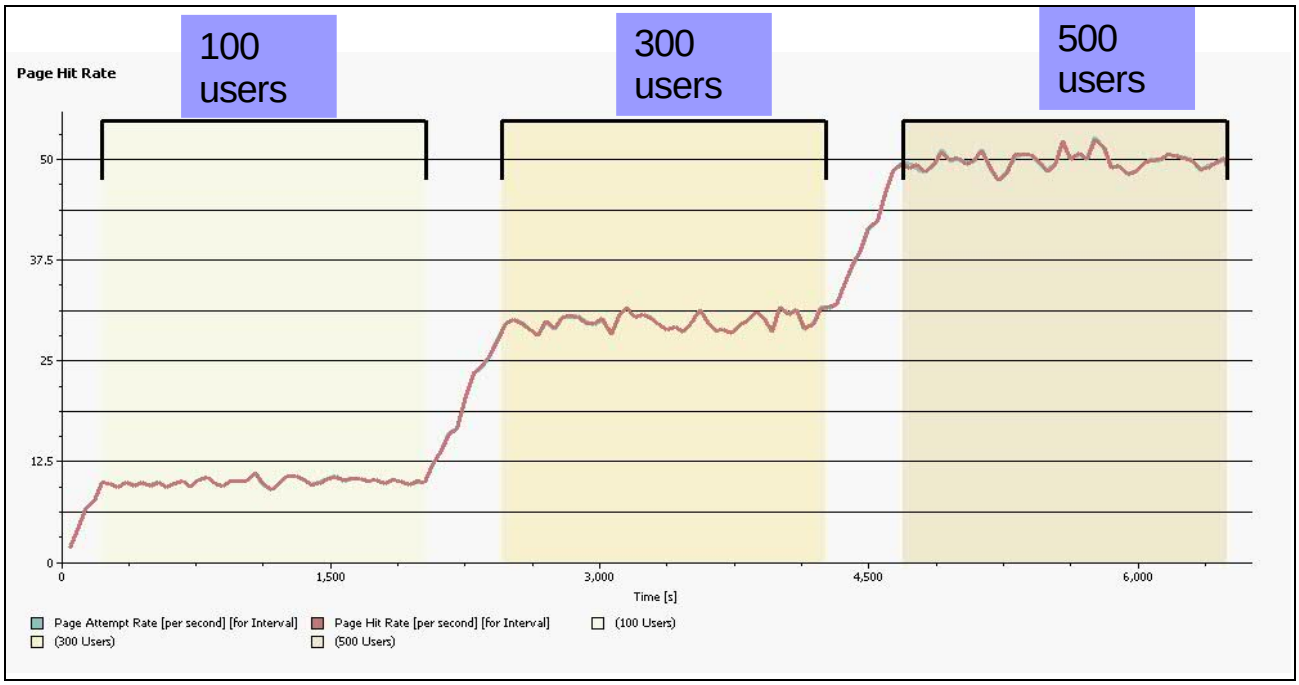

Figure 13. Dynamic Workload Console input stress at different stages (100, 300, 500 concurrent users)

4.1.1.2. Results

Some key performance indicators were used to compare Tivoli Workload Automation V 8.6 FP1 with V9.1.***Table 3*** shows the percentage of improvement.

Category Test Scenario		Response Time Improvements V9.1 vs V8.6 FP1		
		100 users	300 users	500 users
Monitoring	**8.6 AllJobsSUCC(10k)**	6x	12x	21x
	8.6 AllJobsERR(30)	2x	3x	7x
	8.6 AllJSinSUCC(400)	2x	3x	8x
	8.6 AllJSinWAIT(100)	1x	2x	6x
Graphical	**8.6 GraphImpactView**	2x	2x	20x
	8.6 GraphJSView	1x	1x	2x
Modeling	**8.6 CreateJobs_JS**	2x	2x	15x
	8.6 ModifyJob	2x	2x	3x

Table 3. Tivoli Workload Automation V9.1 – V8.6 FP1 key performance indicator comparison

Figure 14 shows the percentile distribution of server response times while increasing the number of

concurrent users.

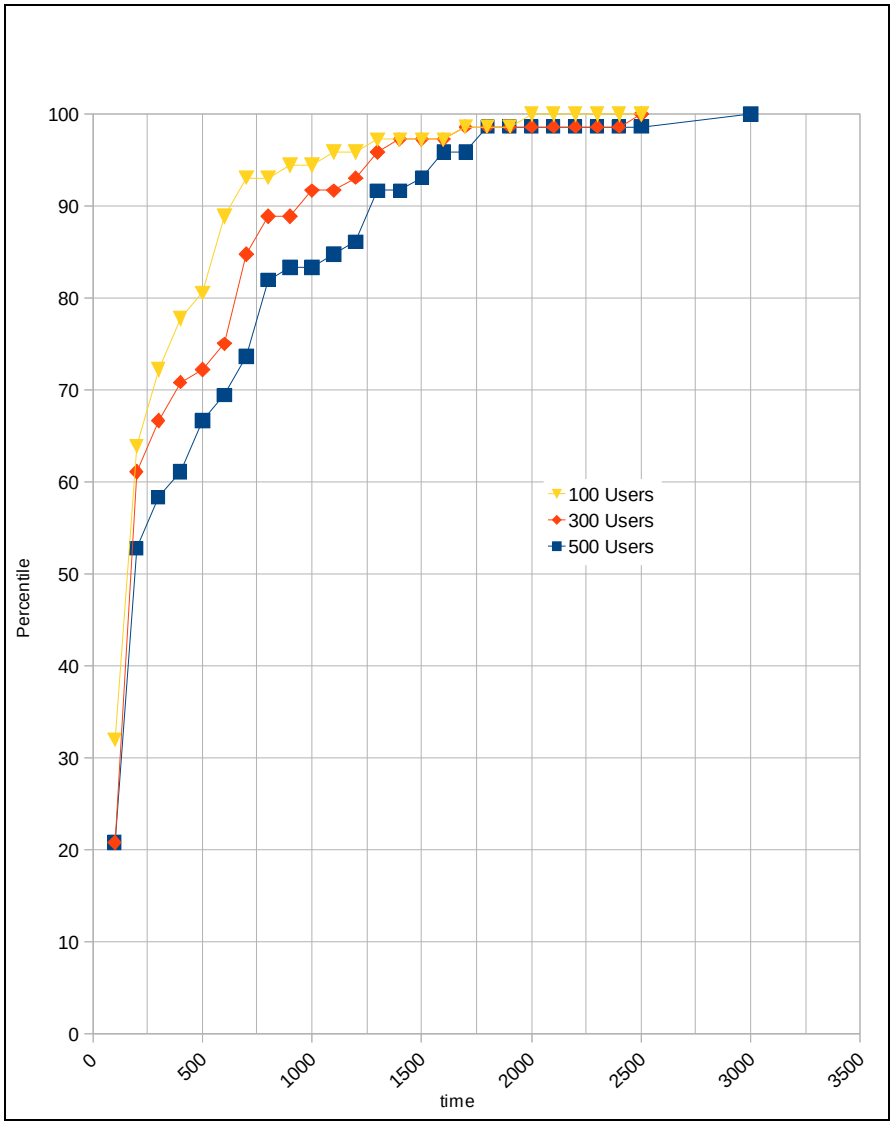

Figure 14 Percentile distribution of response times

Figure 15. Performance improvements shows the trend of response times while increasing the number of concurrent users. Two main correlated results can be underlined:

- Tivoli Workload Automation V9.1 response time is less impacted by concurrency
- Tivoli Workload Automation V9.1 dynamic workload console response time range is almost one order of magnitude less than for the previous release.

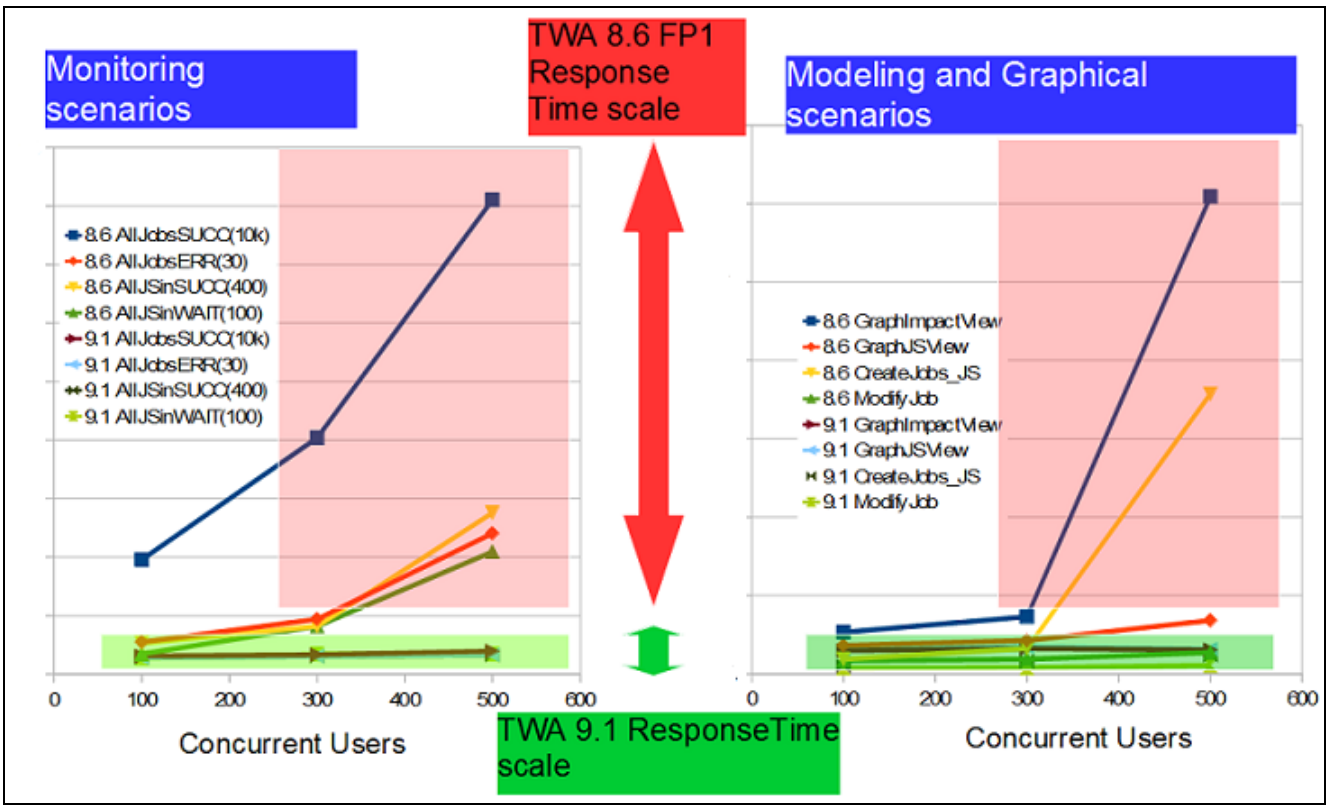

Figure 15. Performance improvements

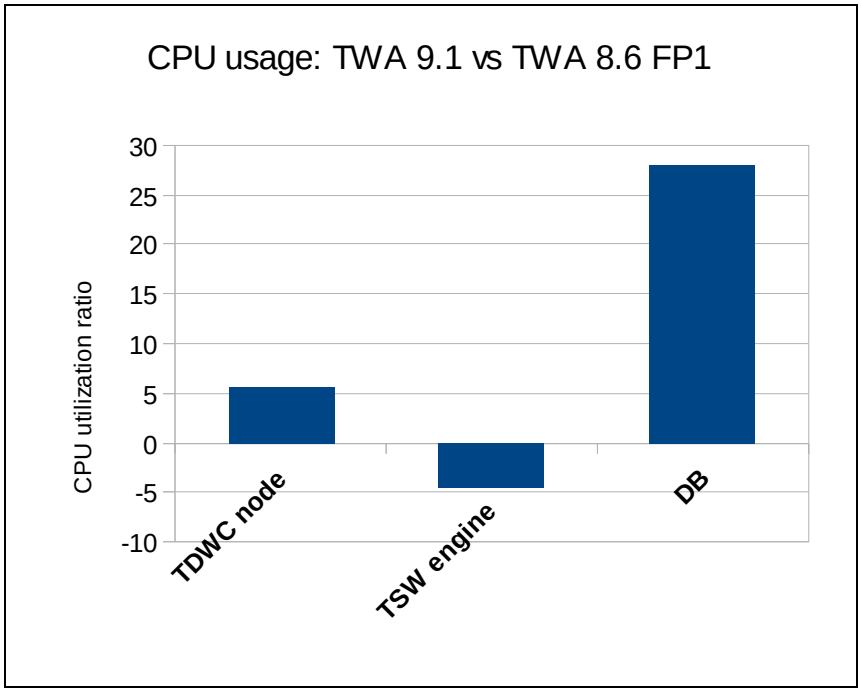

Figure 16. CPU utilization ratio Tivoli Workload Automation V 9.1 vs V8.6 FP1 (negative value indicates less use)

Figure 16 shows the compared resource usage (500 concurrent users test) with respect to the previous release. Note that the mirroring rework moved activities from the engine to the database causing a reduction (see negative value in Figure 16) of CPU usage on the engine node. The high database cpu utilization ratio must be related to the very low usage percentage in previous releases.

4.1.2. Single user graphical activity

Additional tests were made for graphical activities:

- Plan view
- Impact view

The 500-user test graphical scenarios were focused on high concurrency with the job stream and impact views. In this scenario the plan view is run against a plan with 50 job streams and 100 mutual dependencies (Figure 17a). The impact view and relative navigation is run against a three-layer job stream successor (Figure 17b).

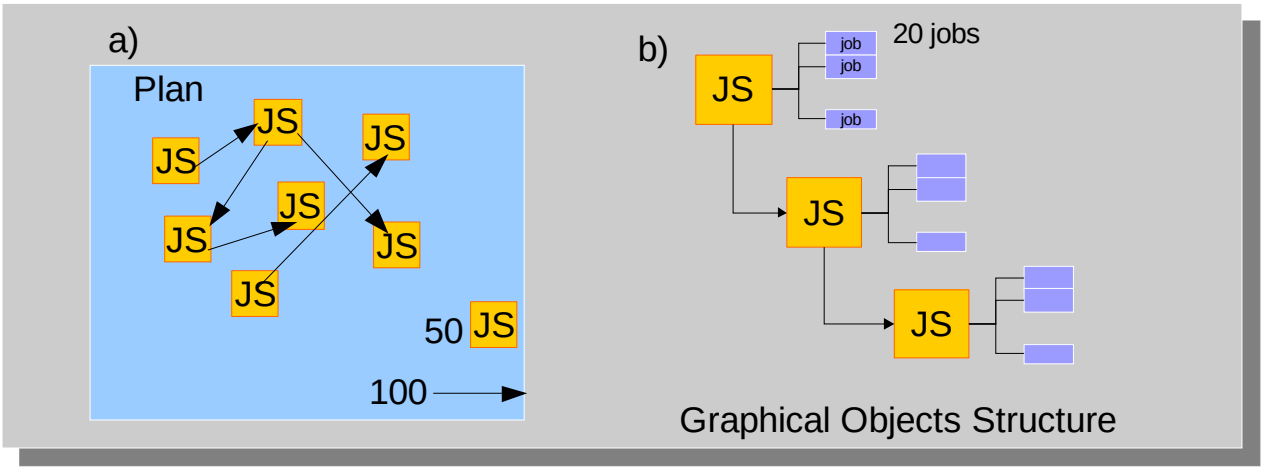

Figure 17. Object structures used in Single User Graphical Activity scenario

4.1.2.1. Workload

A single user runs graphical activities while 100 concurrent users run the monitoring scenario querying 500 job streams in a 100K-job plan with a load of 45 transactions per hour per user.

4.1.2.2. Results

Complex data structure views and navigation are not impacted by concurrent monitoring activities maintaining system responsiveness.

5. Tivoli Workload Automation V9.1 UI concurrency in a long run

5.1. Scenario

In the context of long-run scenarios, the previous test was extended to verify Tivoli Workload Automation V9.1 constancy of performances and reliability emulating a production context. A seven-day workload was applied without stopping any server. Four additional agents were used, two fault-tolerant and two dynamic agents, to run scheduled jobs. Stress on Tivoli Workload Automation was based both on plan execution and Dynamic Workload Console workload with the following specifications:

- Daily Plan (JnextPlan) run every day at10 a.m. including:

  Scheduled 2K job streams in plan containing 126K jobs

  Scheduled 2K job streams submitted runtime using event rule containing 126K jobs

  Job types:

  75K dynamic jobs (35% Java 65% native)

51K Fault-tolerant agent jobs

- Daily Dynamic Workload Console UI activity:

  Mixed UI workload with 300 concurrent users balanced on 5 Dynamic Workload Console nodes at 30 transactions per hour each.

5.1.1. Workload

Plan execution was designed to run for 22 hours every day after JnextPlan completion. The average throughput for scheduling activity was 120 jobs per minute.

Dynamic Workload Console workload was composed of the same mixed scenario as described in the 500 concurrent users test, with the following differences:

- 300 concurrent users
- 30 transactions per user per hour (with a creation of 360 job streams and 720 jobs per hour)
- Query result set for monitoring scenarios was not constant because of dynamic scheduling context

5.1.2. Results

No resource usage degradation was noted during the long run execution and no leaks were detected also at the JVM level.

Figure 17 shows the response time behavior on the last 5 execution days. Modeling activities are the only activities impacted by the execution duration. This can be referred to by the increasing numbers of jobs and job streams whose tables are constantly populated by a rate of $1.5*10^4$ and $8*10^3$ objects per day

To reestablish optimal performances also in the modeling subscenarios, DB2 runstat (which in this context was set to automatic) and reorg commands should be issued (to defragment space and improve I/O access).

Tivoli Workload Automation provides tools to perform such operations (see the Tivoli Workload Automation V9.1 “Administration Guide”, section “Administering the DB2 maintenance feature”).

After running RUNSTAT and REORG, the modeling response times improve consistently (red column).

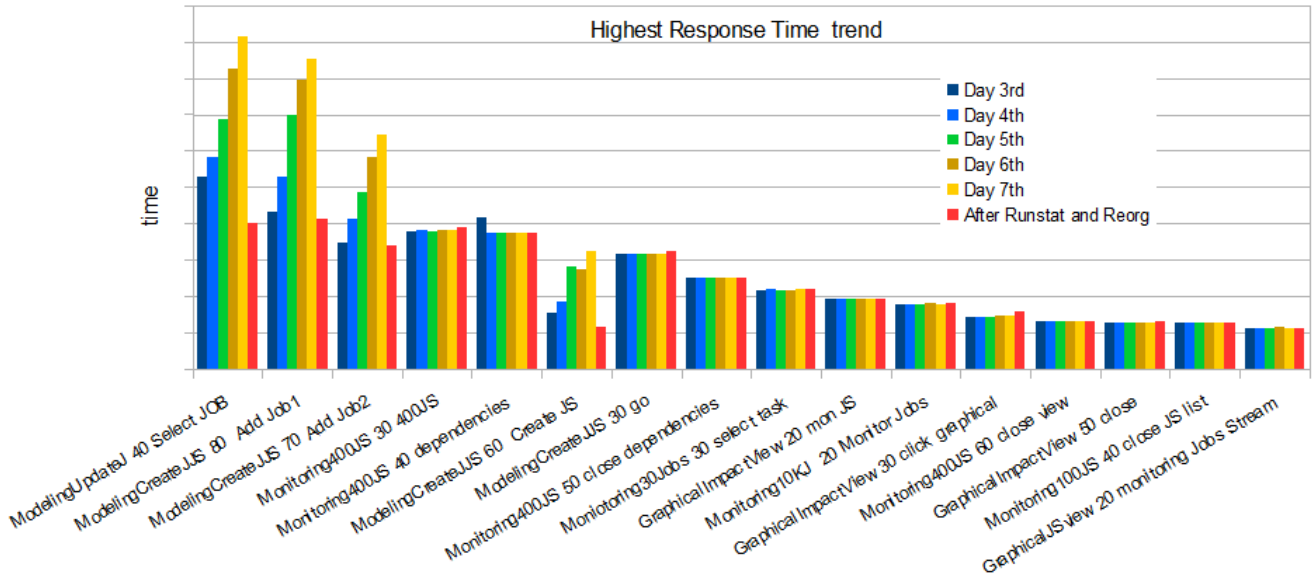

Figure 18. Last 5 days response time trend

6. 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.

6.1. 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.