

**WORK
CENTER
MANAGER**™
...built to change!

Prepared by:
ServiceSPAN
in cooperation with
Sun Microsystems

Application Load Test

Benchmarks of Work Center Manager Orchestrated with
Java CAPS on T2000 Hardware
December 2007

TABLE OF CONTENTS

EXECUTIVE SUMMARY	3
PURPOSE	3
INTENDED USE OF THE BENCHMARKS	4
ACKNOWLEDGEMENTS	4
USING JAVA CAPS WITHIN WCM	4
ABOUT THE PROCESS BEING TESTED	5
LOAD TEST ENVIRONMENT	6
TEST METHODOLOGY	7
TEST RESULTS	8
CONCLUSION	12
ABOUT SERVICESPAN	12
APPENDIX A – JAVA CAPS EINSIGHT BUSINESS PROCESS	13
APPENDIX B – ESTIMATING OTHER BUSINESS PROCESSES	17

Executive Summary

Sun approached ServiceSPAN leveraging the Sun/ServiceSPAN partnership to perform a load test of both Java CAPS and Sun hardware. ServiceSPAN's *Work Center Manager™* was selected due to its use of Java CAPS and its market success.

The load test set out to quantify the amount of CPU processing power required to process a range of simulated workloads, and to quantify the resulting throughput experienced for each simulation. The workloads were based on ServiceSPAN's existing workloads found in deployments of *Work Center Manager™*.

Although ServiceSPAN's existing customer deployments are primarily on mid-range servers; Sun requested the load test to be performed on a T-2000 server(s), which would provide Sun additional data points for the extraordinary performance of Java CAPS on this entry level server.

Among the findings from the load test it was learned that; the WCM application did not experience any noticeable performance penalties due to the Java CAPS orchestration engine, the T-2000's throughput thrived in an environment where significant concurrent transactions were provided, and the T-2000's performance was linear up through 119 transactions per minute, representing 57,000 transactions per day.

Using *Appendix 'B'* of this paper, it is intended that a prospect of Java CAPS would be able to estimate the complexity of their potential Java CAPS processes against a description of the processes used in this load test, then use the results of this load test as a guideline for determining how that potential process would perform on a T-2000.

In 2007, ServiceSPAN performed a similar load test with Sunfire V890 servers. For a copy of that load test go to

http://www.servicespan.net/library/whitepaper/WCM_Load_Test_Results_031807.pdf

Purpose

The purpose of this paper is to provide the results of a joint effort between Sun and ServiceSPAN. The effort was to benchmark both processing power and transaction volumes when using: Sun's Java CAPS, Sun's hardware and ServiceSPAN's application *Work Center Manager™*.

Work Center Manager™ is a purpose built composite application that provides a configurable infrastructure for the management, presentation and resolution of work items—significantly enhancing event driven human workflow. *Work Center Manager™* is deployed in enterprises today, reducing the costs of back office operations.

Intended Use of the Benchmarks

Using Appendix 'B' of this paper, it is intended that a prospect of Java CAPS would be able to estimate the complexity of their potential Java CAPS processes against a description of the processes used in this load test, then use the results of this load test as a guideline for determining how that potential process would perform on a T-2000.

Acknowledgements

Many thanks to the Sun team listed below, (as well as to those not listed) for their assistance in completing this load test. ServiceSPAN is appreciative of the opportunity to help further the adoption of Sun Java CAPS and Sun technology in marketplace.

- Alexis Roos
- Murali Pottlapelli
- Fred Aabedi
- Ross Altman
- Desten Broach
- Theresa Villatore-Silva

Using Java CAPS within WCM

ServiceSPAN uses Java CAPS for its EAI components and BPM capability. Java CAPS also helps illustrate WCM relevance as an SOA based application. Java CAPS' integration centric capabilities complement WCM's human centric capabilities providing a world class application with a strong ROI.

About the Process Being Tested

The process is a back-end process that connects to a customer's mainframe to capture incoming business events. In this case the events are new and changed trouble tickets. The tickets are run through a pair of SOAP services that recognize if the ticket is new or changed, and registers the tickets into a database where they are enriched for presentation. Users are presented the tickets via a desktop browser.

The Java CAPS eInsight business process created for this test is summarized below. A more detailed explanation and screen shots are included in Appendix A of this paper.

1. A 3270 based trouble ticket document, screen scraped using 3270 emulation, is passed into the business process. This unstructured document contains 7,600 characters with many fields unlabeled and some necessary information embedded in comments.
2. A SOAP service performs a document characterization (Parsing) process, that returns fields from the document after running a series of regular expression based business rules against the text.
 - a. This CPU intensive use of regular expression based parsing is useful for source documents that are unstructured and is flexible to shifts in field position and finding information in free formatted fields like comments.
 - b. For structured and unstructured documents alike, this service can also derive information through lookups to configuration tables and comparisons between properties.
 - c. There are over 30 fields that are returned from this document that will be used to determine the priority, route the document to people, or used in some manner to reduce the time required to work the trouble ticket.
3. Business logic is added to handle oddities in the mainframe system specific to this customer, such as the document or customer id being blank.
4. A second SOAP service, called "registration", stores the ticket and all the parsed fields into the Oracle database and assigns default values in the event a field(s) are not found on the 3270 screen.
5. Finally, seven additional SOAP services are called in parallel, triggering mainframe based research requests and queueing the responses for later processing by a series of scripts. (The running of the scripts was excluded from our load test, as we had no reasonable substitute for these mainframe hosts available)

Load Test Environment

The load test was performed with a standard configuration of SUN T-2000 servers. ServiceSPAN installed the hardware, Solaris 10 OS, and Oracle Database in a default configuration. Sun's OEM Engineering Team recommended a Java CAPS JVM configuration be used as well.

The environment consisted of four SunFire T-2000 servers each with a single 8-core 1.2 Ghz processor, equipped with standard disk and memory configurations, and Solaris 10.

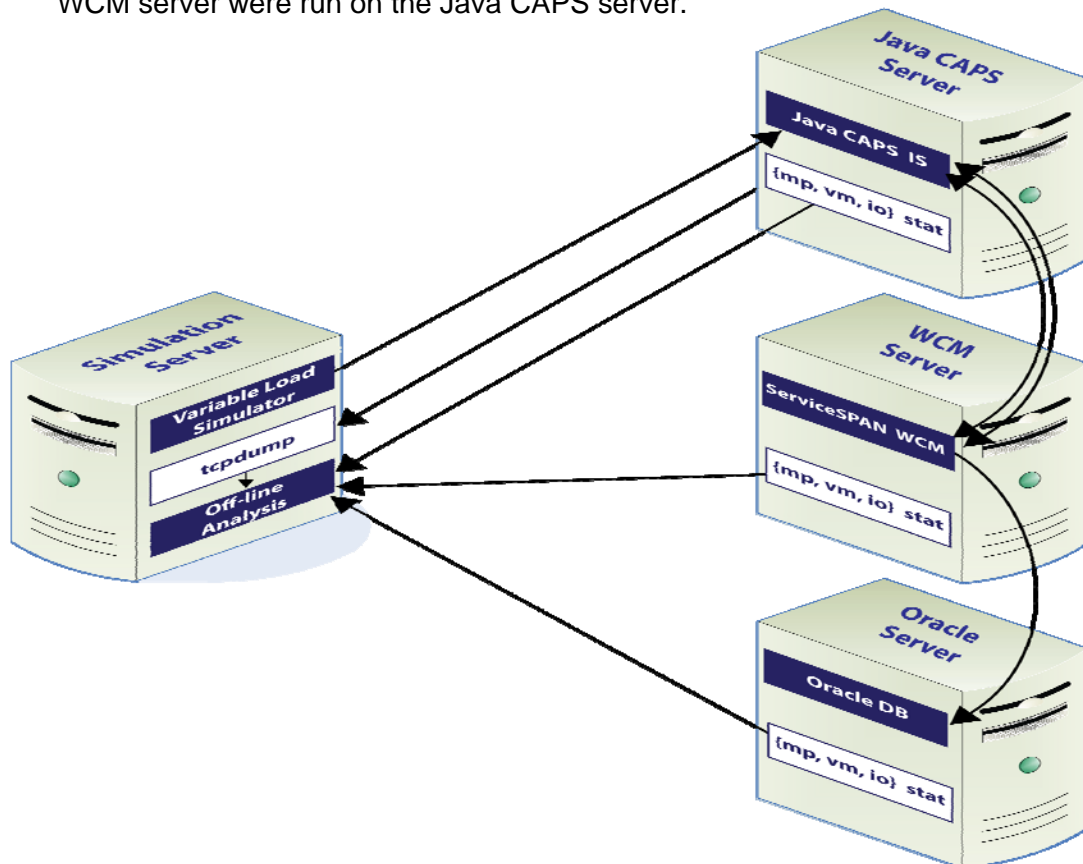
The servers were identified as follow:

Benchmark Support Servers:

1. Simulation Server – Software to emulate a mainframe trouble ticketing application offering up new and changed trouble tickets
2. Oracle Server – Oracle 10g in default out of the box configuration (no optimization).

Production servers to benchmark:

3. Java CAPS Server – Java CAPS elnsight 5.1.3 configured with the business process contained in Appendix A
4. WCM Server – WCM Version 1.6.3 software services configured as in production, running in a Solaris Zone. During one running of the load test, the functions of the WCM server were run on the Java CAPS server.



Test Methodology

The test was conducted as follows:

1. The variable load simulator utility maintains a predefined number of concurrent sessions with Java CAPS integration server. Each session was running in synchronous mode with the simulator triggering a new instance of the business process each time one of the existing instances completed. The number of concurrent sessions started at 4, and was increased until a point of diminished return was reached.
2. The tcpdump utility captured HTTP traffic between Java CAPS and WCM for offline analysis
3. The mpstat, vmstat, and iostat utilities captured OS counters on each host for offline analysis
4. The test was repeated by incrementing the number of sessions by 2.

Test Results

The test results are offered in two ways. First as a single server on which both Java CAPS and the WCM services are running (*Combined Configuration*), and then separately with Java CAPS and WCM running on separate servers (*Separate Configuration*) where the individual cost of CPU resources could be reliably measured.

Combined Configuration: Single server with both Java CAPS and WCM

In the table to the right, the results are displayed for the combined configuration. Concurrent sessions increased from 4 to 30. Disk I/O, network I/O, and database server CPU were all negligible. Database disk I/O reached a maximum of 20% at 120 trx/min, demonstrating that this process is CPU bound.

Each session repeatedly executed a single Java CAPS insight business process, or transaction, one at a time as fast as possible. Each transaction included orchestrating 9 SOAP calls.

Concurrent Sessions	Trx / Min	Combined Latency (seconds)	Combined CPU (%Util)	SOAP calls / minute
4	30	7.8	14%	270
6	45	7.9	20%	405
8	59	8.1	27%	531
10	70	8.5	34%	630
12	81	8.9	40%	729
14	91	9.3	47%	819
16	98	9.7	53%	882
18	104	10.3	60%	936
20	107	11.2	66%	963
22	110	11.6	73%	990
24	112	12.4	79%	1008
26	116	13.4	85%	1044
28	119	14	90%	1071
30	120	15.5	95%	1080

Note: The latency is rather high for this service due to the use of regular expression business rules that allow for a great amount of flexibility. A different CPU would run this process faster, but the T-2000 in this application scales up wonderfully and is very well suited for the volumes required for this background processing task.

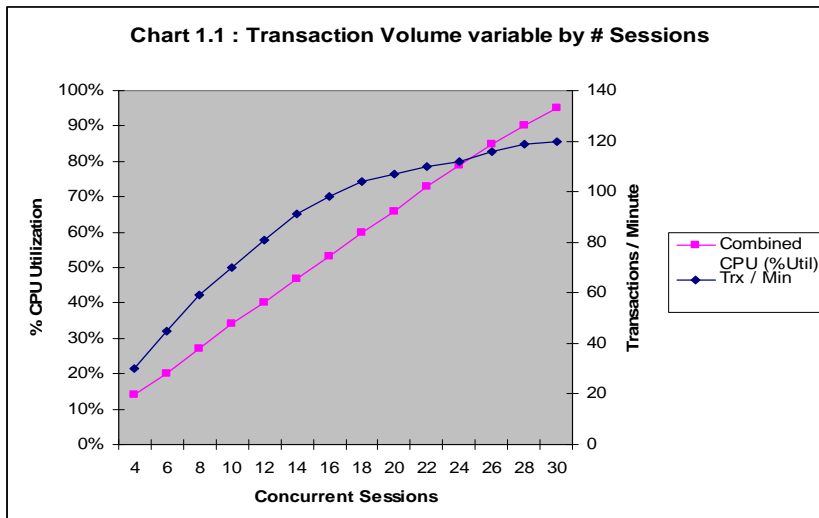
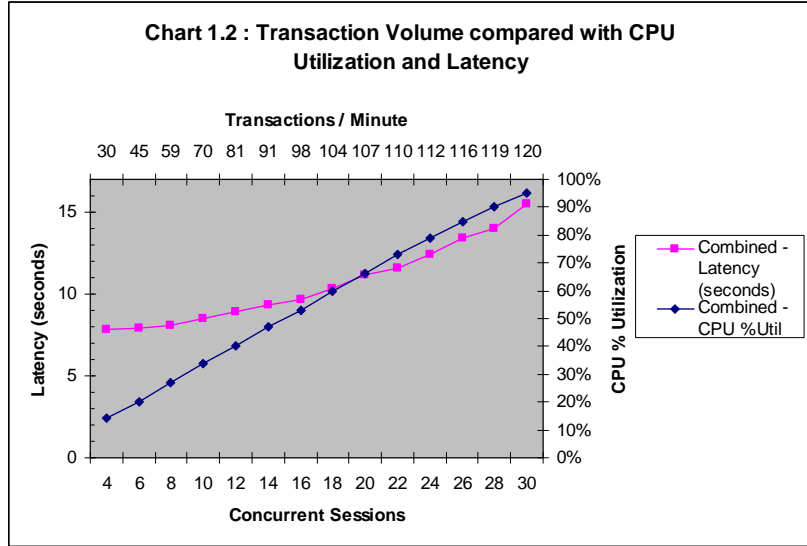


Chart 1.1, shows that the starting number of 4 concurrent threads is engaging only a small portion of the CPU. The large number of threads of the T-2000 CPU are being used to great benefit as the number of concurrent sessions are increased.

In Chart 1.2 we can see that the additional concurrent sessions are added with little increase in latency. On the left side of the graph a 200% increase in sessions generates a 4% increased latency and a net result of almost 200% increase in transactions. Further up the curve, presumably as most of the CPU threads are in use, an increase in latency remains linear as it slowly increases up through 90% of CPU utilization.



But How Many Users?

The data presented above was measured in transactions, latency seconds, sessions, etc.; but what about the number of users? In the real life scenario of the process used in this load test human beings (users) would be using the application too.

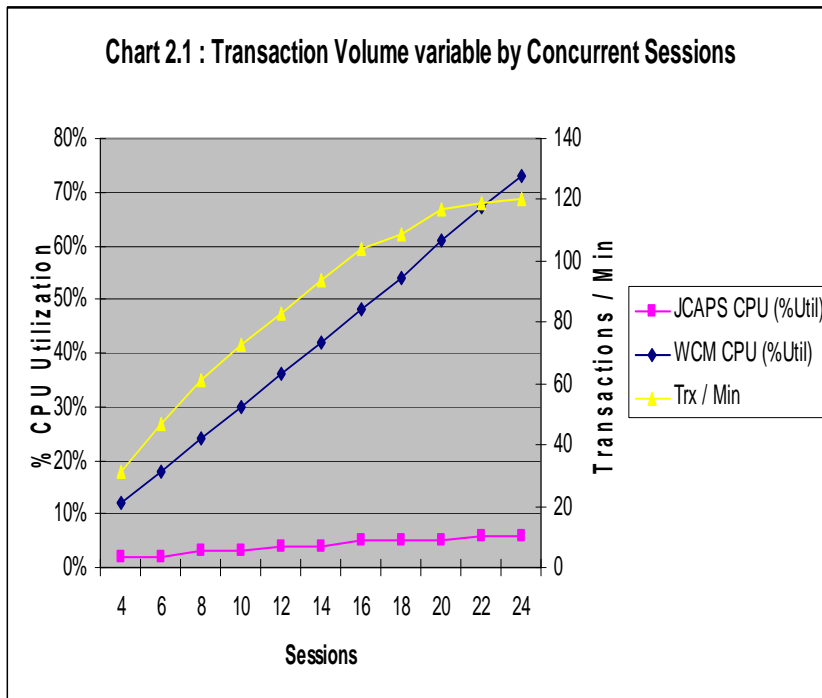
So how much CPU would be required to support 100 users of the business process tested in this paper—**very little**. In fact, only an impressive 14% of the CPU would be required for both Java CAPS and WCM combined.

(Note: 100 users would equate to 33 transactions/minute based on 3 minutes per transaction)

Separate Configuration One Server with Java CAPS, One Server with WCM

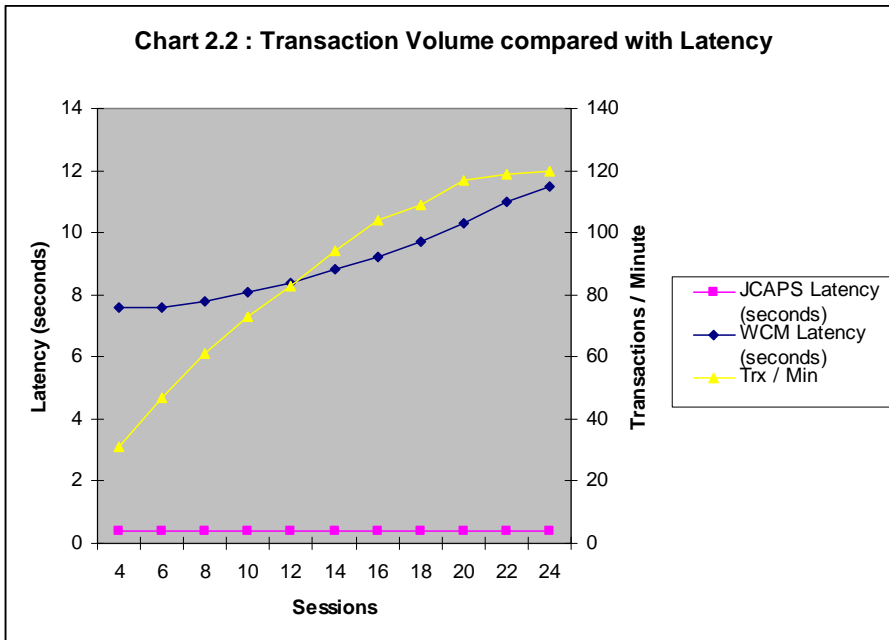
In the table to the right, the results are displayed for the separate configuration. Concurrent sessions increased from 4 to 30. Disk I/O, network I/O, and database server CPU were all negligible. Database disk I/O reached a maximum of 20% at 120 trx/min.

Concurrent Sessions	Trx / Min	JCAPS Latency (sec.)	JCAPS CPU (%Util)	WCM Latency (seconds)	WCM CPU (%Util)	SOAP calls per minute
4	31	0.4	2%	7.6	12%	279
6	47	0.4	2%	7.6	18%	423
8	61	0.4	3%	7.8	24%	549
10	73	0.4	3%	8.1	30%	657
12	83	0.4	4%	8.4	36%	747
14	94	0.4	4%	8.8	42%	846
16	104	0.4	5%	9.2	48%	936
18	109	0.4	5%	9.7	54%	981
20	117	0.4	5%	10.3	61%	1053
22	119	0.4	6%	11	67%	1071
24	120	0.4	6%	11.5	73%	1080



In chart 2.1 we see the results of running Java CAPS and WCM SOAP services on separate T-2000's shows:

- Java CAPS uses a much smaller amount of the processing power of the T-2000 than the process being orchestrated here.
- With Java CAPS utilizing a peak of only 6% of the CPU, this single T-2000 could orchestrate an entire farm of 15+ Sun servers running the business process used here.

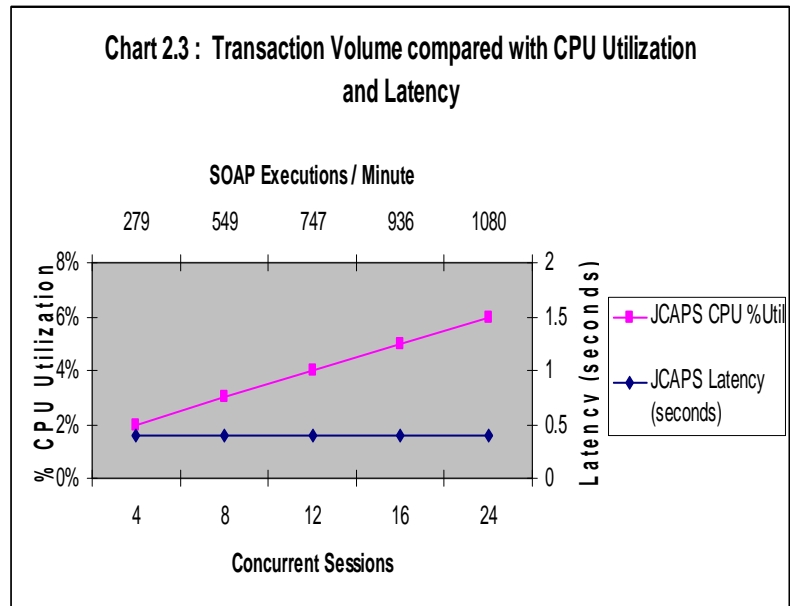


As shown in chart 2.2, Java CAPS is also responsible for a small amount of the latency, holding stubbornly to a 400ms cost as the number of sessions increase.

Amplified in chart 2.3, at this low range of CPU utilization, Java CAPS admirably held steady at 400ms latency as the number of concurrent sessions increase.

The number of SOAP executions per minute has been added. This was calculated by taking the number of transactions per minute x 9 SOAP calls per transaction. Two of the SOAP calls are non-trivial, passing a total of 30 data elements including date, time, string and a 7,000 character document.

Be reminded that these are not Java CAPS performance numbers in isolation. As impressive as these real-world examples may be, actual SOAP execution times could be much shorter if we had simpler services with fewer than 30 data elements to marshal and un-marshall through WSDLs. Similarly, if we didn't have a mix of seven simple SOAP executions to offset the high complexity of our first two services, our execution times would be longer.



Conclusion

- The Java CAPS orchestration engine did not introduce any noticeable performance penalty even when offered 30 simultaneous transactions. For installations of this size, adding orchestration capabilities to the system is unlikely to require significant changes in capacity planning.
- T2000 server, due to its hardware parallelism, achieves most throughput in an environment which can provide significant numbers of concurrent transactions. Java CAPS running alone or sharing the T-2000 with the services to be executed, demonstrated performance consistent with efficiently utilizing the multi-threaded nature of the CPU
- The T-2000 server's performance was well-behaved and linear up through 119 transactions per minute, which could support as many as 57,000 daily transactions of this type which included complex document parsing and storage over an 8 hour period. If these manual transactions required 3 minutes of manual activity each, that would be enough capacity to support 400 employees with 7 hours of productive time per day in one shift, or 1200 employees with daily productive time across 3 shifts.
- Our database server reported negligible CPU utilization and 20% disk utilization. If a customer's requirement was lower than 119 transactions per minute, it would be a reasonable expectation that the Oracle database could be co-located with Java CAPS and WCM without significant performance penalty.

About ServiceSPAN

ServiceSPAN provides work center automation software to improve system-to-human and human-to-human workflow. Our software is typically deployed in work centers where reducing work item handling time, distributing work efficiently, increasing information flow and simplifying user interfaces are key components of efficient operations.

Work Center Manager™ (WCM) is a purpose-built application that provides a configurable infrastructure for the management, presentation and resolution of work items—significantly enhancing event driven human workflow. *Work Center Manager™* is deployed in enterprises today, reducing the cost of back office operations.

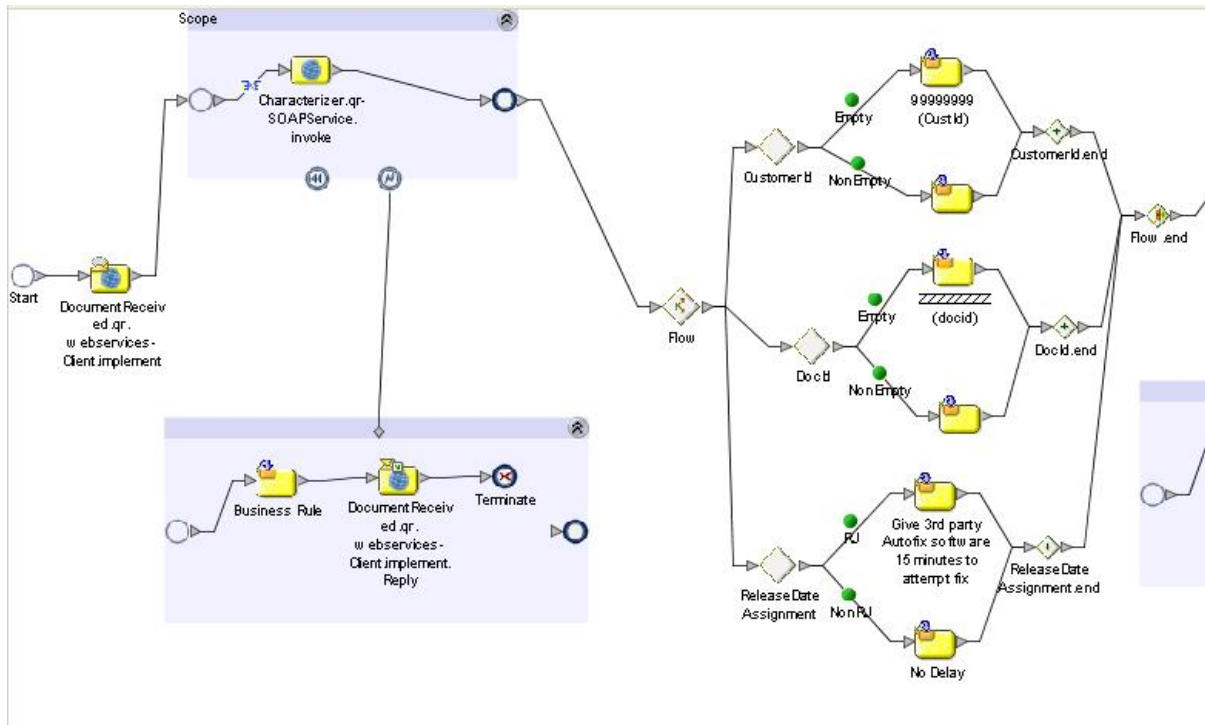
WCM creates value in a work center by focusing on sub-task automation. Work center tasks are comprised of sub-tasks such as; looking up information, cross referencing data, interacting with other organizations, updating data entry values, etc. The cost of the "task" is a product of the time and resources (labor and systems) required to perform the task. Sub-task automation, directly reduces the cost of a task, by reducing the amount of time and resources required to perform the task. As the percentage of sub-task automation increases so does the percentage of savings, along with WCM's value and the ability to implement Zero-Touch initiatives.

Customers have found that WCM's value proposition of improving work center productivity, reducing operating costs, and facilitating faster service delivery have driven a strong ROI and improved their bottom line financials.

For additional information about this report or Work Center Manager:

ServiceSPAN
101 Sunnyside Blvd.
Plainview, NY 11803
(516) 576-8000 www.servicespan.net

Appendix A – Java CAPS elnsight Business Process used for Load test



Java CAPS elnsight Business Process, Page 1

This business process executes from left to right.

This business process supports multiple manual work sources which consist of a document based work request.

In the production environment in which this business process is used, WCM provides a browser based front-end to a complex enterprise-wide trouble ticketing system for the purpose of enabling this specific group of users the ability to perform skills-based routing, perform a data augmentation function before user display and the ability to measure and manage the manual work with capabilities that are cost prohibitive to obtain from the original application. The end benefit is a decrease in the time necessary to process each work request.

This business process begins with the receipt of a trouble ticket from our simulator. Each trouble ticket is a series of 3270 screens captured with a 3270 script that searches for new and changed trouble tickets. Each ticket consists of 7,600 characters of text.

“Characterize”, the first SOAP call to WCM, is responsible for parsing the 3270 trouble ticket into properties that can be used for the rest of the business process and SOAP calls to WCM. As everything else executes very rapidly, this one SOAP call is responsible for 90% of the measured latency in this business process.

Sample Document being processed

```

MSCR MC 401 SCREENER EC 451 17238 01-01-06 0319A
TN TTN 0000001 SELECT STATUS PS RTE RETURN

STATUS: PS 001 01-01-06 0318A /HNDL RCMAC - ROUTE TO RCMAC
TTN: 0000001 TN: 605 5555555 COMM: 11-23-07 0400P REACH: 9999999991
DOE, JOHN ACCESS: A B
1000 S DAHLIA ST, GL CS: RES SC: 1FR RTE: 303 691 1199
APT I110 LAST CLRD: 03-29-05
TROUBLE: CCO CCO LD BLOCK ON LINE REFD BO PLEASE RMV SCG3X PD 135.35
VER: 6 SUMMARY: SPEECH CTTN:
SCRN RSLT: 900 NARR:
FAC CABLE AIN LINEBACKER ASSOC AIN SVC
F1 29 1273 0573 X 4815 E MISSISSIPPI AV
F2
FZ 4815EM 0767 0042 IT 1086 S DAHLIA BLDG I
OE: AA05-0-09-14
EST
WP IST RTE RSLT 01-01-06 0319A OOS FL1 FL2 FL3
NARR
FST
T D C FL1 FL2 FL3 X
NARR
TEST/INFO RBOR BROADCAST
233

MSCR MC 948 SCREENER EC 451 17238 01-01-06 0319A
TN TTN 0000001 SELECT STATUS PS RTE RETURN

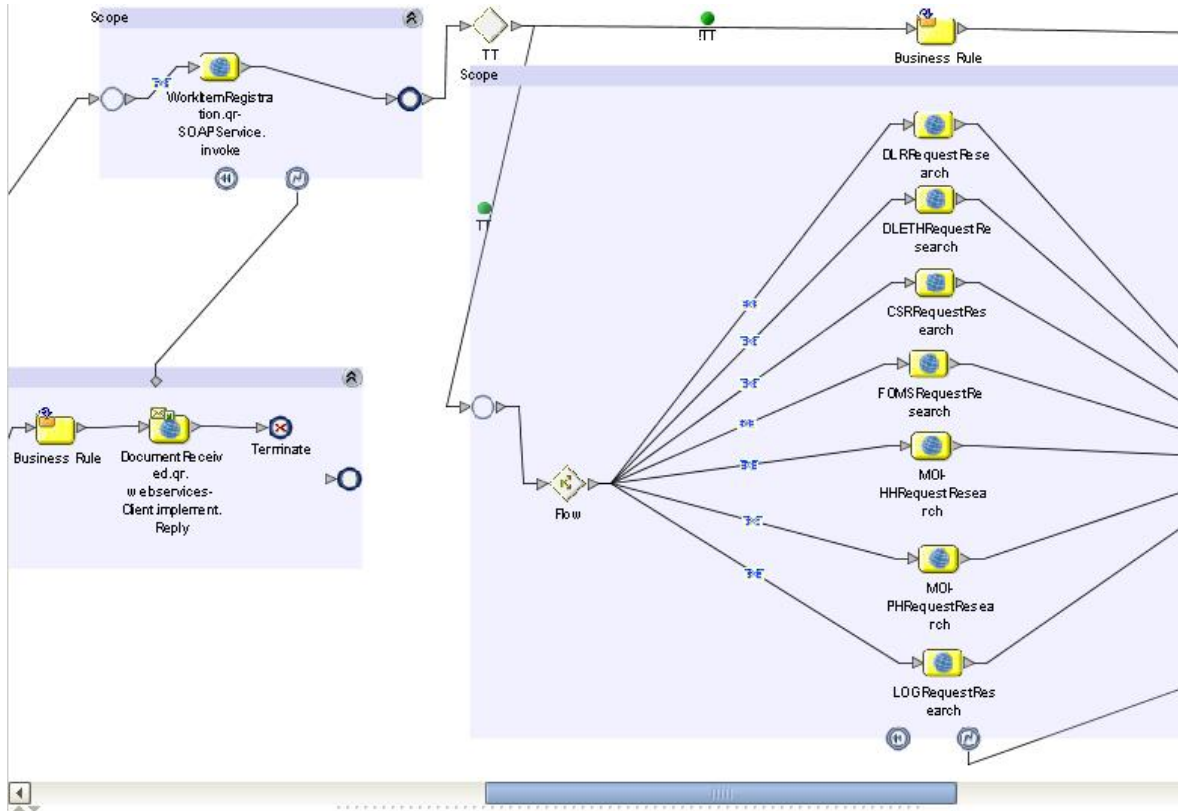
STATUS: PS 001 01-01-06 0318A /HNDL RCMAC - ROUTE TO RCMAC
TN UNIT SUB STAT OOS DATE/TIME
3035555555 82800000 0 6 N 01-01-06 0318A
EC WP IST RTE RSLT DATE TIME
299 0 PS 04700001 000 01-01-06 0318A
NARR: /HNDL RCMAC - ROUTE TO RCMAC
299 0 RSC 82800000 000 01-01-06 0318A
NARR: /HNDL RCMAC - ROUTE TO RCMAC
299 0 PDI 00000911 900 01-01-06 0318A
NARR: PENDING SAB TO UNIT 82800000
299 1 PDI 00000998 900 01-01-06 0318A
NARR: /HNDL RCMAC - ROUTE TO RCMAC
299 0 PSH 00000299 6 01-01-06 0318A
EST
WP IST RTE RSLT 01-01-06 0319A OOS FL1 FL2 FL3
NARR
FST
T D C FL1 FL2 FL3 X
NARR
TEST/INFO RBOR BROADCAST
233

```

The decision logic on the right side of the **Java CAPS insight Business Process, Page 1 (previous page)** takes care of three special cases of trouble tickets. None of these however, occurred with the data provided in the simulation:

- Missing Customer ID, substitute 99999999
- Missing Trouble Ticket #, substitute 9999999
- Give an automatic fixing system, external to this business process, a 15 minute window to attempt to automate the work.

On the bottom left side of the **Java CAPS insight Business Process, Page 1 (previous page)** is an exception business rule to throw an error if the ticket could not be processed.

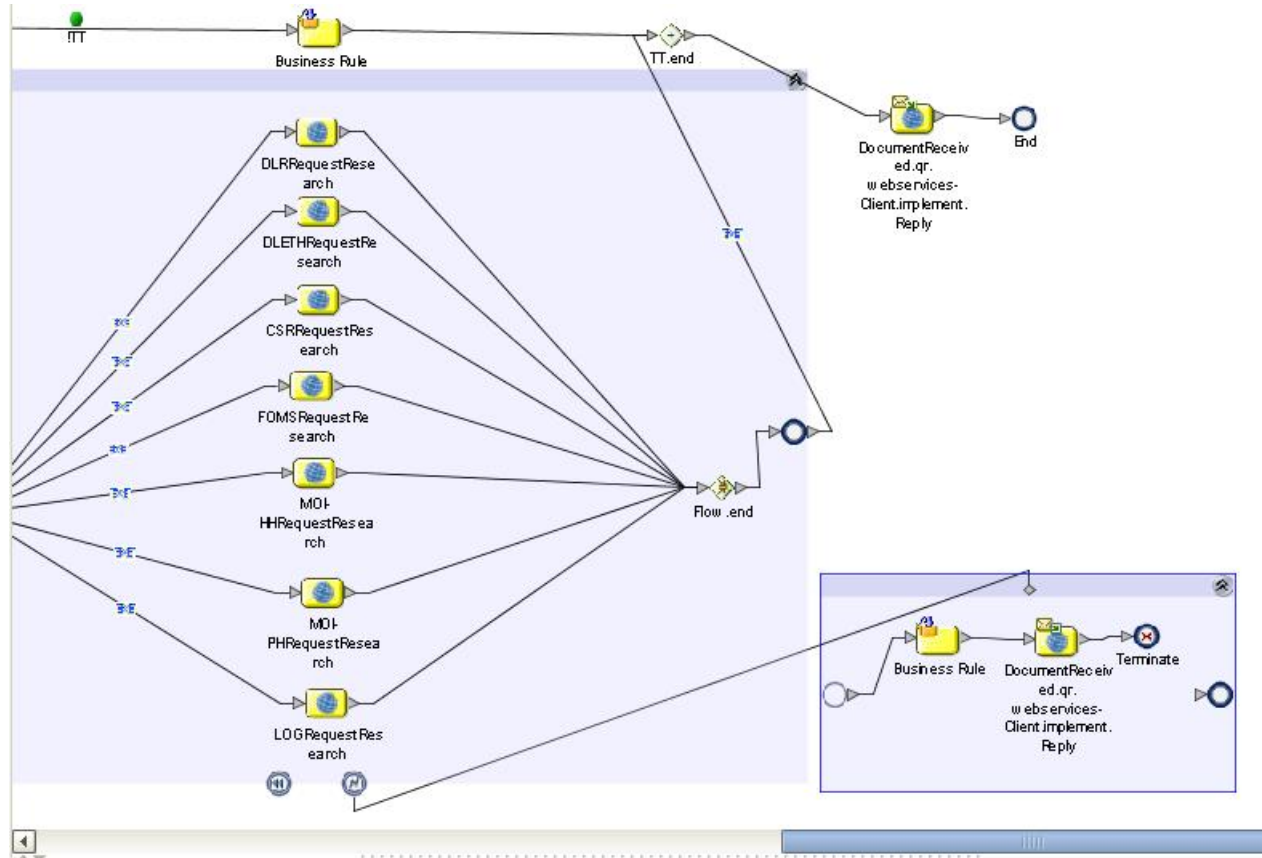


Java CAPS insight Business Process, Page 2

Work Item Registration – This SOAP service is responsible for registering (saving) the trouble ticket in Oracle, where it will be under the control of application settings (priority, work distribution, etc.) that are overseen by local management.

On the right, and providing this is a valid trouble ticket, research requests will be queued for each of these seven hosts. These 7 SOAP calls run in parallel. If this were running in a production environment, another 3270 script would execute these requests against another set of mainframes, and the recorded results would be co-presented with the trouble tickets to users.

On the bottom left is an exception business rule to throw an error if the ticket could not be stored.



Java CAPS insight Business Process, Page 3

Then this business process ends, while the trouble ticket and its future research are persisted in Oracle where it will wait until a user chooses “AUTO” to be presented the next highest priority work that matches their skills and assignment.

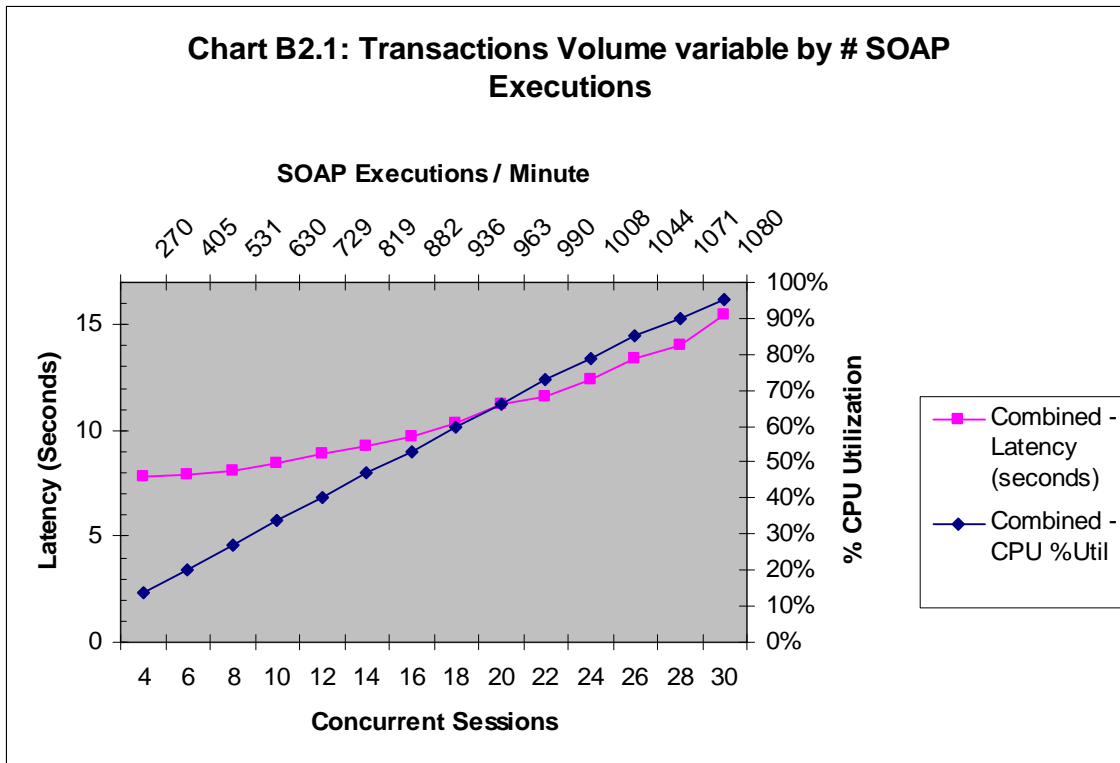
On the bottom right is an exception business rule to throw an error if the ticket research could not be queued.

Appendix B – Estimating Other Business Processes

This Appendix 'B' is for sizing purposes, and intended to give prospects a guideline to evaluate how their real-world process might perform on a T-2000 server.

Concurrent Sessions	Trx / Min	Combined Latency (seconds)	Combined CPU (%Util)	SOAP calls / minute
4	30	7.8	14%	270
6	45	7.9	20%	405
8	59	8.1	27%	531
10	70	8.5	34%	630
12	81	8.9	40%	729
14	91	9.3	47%	819
16	98	9.7	53%	882
18	104	10.3	60%	936
20	107	11.2	66%	963
22	110	11.6	73%	990
24	112	12.4	79%	1008
26	116	13.4	85%	1044
28	119	14	90%	1071
30	120	15.5	95%	1080

The SOAP calls/min column in the chart above is calculated by taking the Trx/min of our back-end business process and multiplying by 9, (the number of SOAP calls in the business process).



The business process in Appendix A consists of 9 SOAP calls.

One of the calls, “Characterizer” (document parsing with regular expression style business rules), represents 90% of the latency in this business process. It is a CPU intensive process with little I/O). On a scale from 1 to 100, this SOAP service is a complexity of 90 as it involves the execution of flexible regular expression based business rules on each execution of this SOAP call.

Another call, “Register”, is responsible for saving the manual work item and all its properties into an Oracle database. This is a reasonably complex Oracle table insertion with three tables involved (including an event history) and many indexes. This represents less than 5% of the latency. On a scale of 1 to 100, this SOAP service is a complexity of 20, as it involves un-marshaling over 30 data fields and creating database rows in three Oracle tables.

The remaining 7 SOAP calls, executed simultaneously, insert a row into a simple Oracle database table that queues requests for mainframe access by geography (mainframe read-only requests). On the same scale of 1 to 100 these calls are a complexity 3, leaving the remaining easiest two complexity levels for extremely trivial SOAP calls.

To evaluate how another real-world process might perform on a T-2000, I propose the following method:

1. Consider the average complexity of the real-world SOAP calls in this document to be an “14.6” on a scale from 1 to 100, as calculated below:

SOAP Service	Complexity	# uses
Characterize	90	1
Register	10	1
Research Request	2	7
Average	14.6	

2. Make a judgment call using the information in this paper, as to the comparative average complexity of a new business process. Call the complexity of such new business process “AC” (Average Complexity)
3. Next determine the number of transactions (business processes completed end-end) needed to be completed every minute. Call this “TPM” (transactions per minute)
4. Next multiply CA & TPM, call this value “SPM” or SOAP executions per minute
5. Find the next highest value along the upper x-axis of the Chart B2.1 (‘SOAP Executions per Minute’), and look up to determine the cost in CPU Utilization to find your answer.

Example: AC=14.6, TPM=50,
 SPM = 14.6 x 50 = 730 SOAP Executions / minute
 On the table 730 is approx 40% CPU utilization

6. Validate if your business process has enough concurrency to achieve this:

Follow the lower X-axis (Concurrent Sessions) to find if the number of concurrent occurrences of your business process that would need to be running in parallel. If you can architect your solution with at least that many, then the result you have in #5 above would appear valid if your values for “CA” and “TPM” are correct.

Example: In this example the lower X-axis (Concurrent Sessions) reveals a need for 12 or more concurrent sessions of this business process running to realize 40% CPU utilization

If your business process is outside the range of these tables and charts, there are a number of server options available from SUN to consider. The T-2000 represents the lower end of the SUN range of servers in terms of raw CPU processing power.

If volume is an issue, look at distributing processing across multiple T-2000's (using one for Java CAPS and the others for your services), or select a SUN server based on the SPARC IV+ architecture such as the SunFire V890 series. The SunFire V890 series similarly has multiple cores and multiple threads per core, but also supports multiple CPU sockets.

If volume is not the issue but concurrency or latency is, then a SUN server with fewer threads but higher CPU speed would be the best path forward.

Comments ???

Please write to me via my Java CAPS blog, to tell me if this has helped. Thanks

<http://javacaps.blogspot.com/>