An Oracle White Paper
September 2014

IBM WebSphere Application Server on Oracle's SPARC T5 Server: Performance, Scaling and Best Practices

Eric Reid and Niting Qi — Oracle Systems ISV Engineering

Kevin Arhelger — IBM WebSphere Performance

Who Should Read This Document

This paper is intended for anyone considering running IBM WebSphere Application Server (WAS) on Oracle's SPARC T-Series servers. For IBM and Oracle customers, this includes systems administrators and system architects; we also encourage Oracle and IBM Sales and Support teams to consult this document.

Executive Summary

The latest release of WAS is shown to perform exceptionally well on SPARC T-Series servers. Specifically

- SPARC T-Series servers provide a scalable, dependable, and performant platform for the deployment of WAS application instances.
- Application architects should have confidence in deploying WAS to multiple "building blocks" consisting of one or more SPARC processors from Oracle; these building blocks may be instantiated as physical or virtualized resources.
- WAS demonstrates 2.7X better throughput on Oracle's SPARC T5 processors than on Oracle's SPARC T4 processors, with improved response time.
- Use of Oracle VM Server for SPARC and Oracle Solaris Zones virtualization technologies not only allows clean separation of virtual environments, but also allows for full and efficient use of physical resources, yielding maximum performance.
- SPARC T-Series servers demonstrate excellent linear scalability for WAS as the number of CPU cores increase.
- Careful physical resource management via Oracle Solaris, when done in concert with the above-mentioned virtualization technologies, can improve performance for WAS instances on SPARC T-Series servers running Oracle Solaris.

Rather than attempt to saturate a single SPARC T-Series server with a single instance of WAS, scaling performance was considered in terms of defining proper "building blocks" for WAS users. By making use of proper resource management and virtualization technologies available in Oracle Solaris and SPARC T-Series servers, optimal overall application scaling can be obtained by the use of multiple instances of WAS—either clustered or non-clustered—each making use of a portion of the available server resources.

This paper also contains a discussion of best practices for running WAS v8.5.5 on Oracle Solaris 11 on SPARC T-Series servers.

Introduction

Four years after the acquisition of Sun Microsystems, Oracle continues to invest heavily in SPARC processors and systems, as well as Oracle Solaris. New products based on these technologies are consistently and regularly being delivered to customers, and Oracle's public SPARC roadmap, publically available on Oracle's website, reflects the ongoing commitment of Oracle to these products and technologies. The link to this roadmap is available in the "References" section.

The five-year trajectory in the roadmap is intended to deliver a tremendous amount of compute power:

- Cores-per-CPU counts are planned to increase by 4x
- Hardware threads-per-CPU are planned to increase by 32x
- Memory capacity is planned to increase by 16x
- Database throughput is planned to increase by 40x
- Java Operations Per Second is planned to increase by 10x

In order to take advantage of the massive amount of parallel processing available in SPARC-based systems, applications must be deployed in a manner that will minimize the time spent waiting for serial resources, including disk I/O, memory, and application synchronization locks. This paper describes a scaling and performance study of an application running on a SPARC T5–based server with twice the number of virtual CPUs as a comparable SPARC T4–based server. The goal, in this environment, is to ensure that the maximum amount of work is being carried out across all these virtual CPUs.

This paper describes a scaling and performance study of IBM WebSphere Application Server Version 8.5.5 running IBM's WebSphere DayTrader 3 benchmark, comparing single-CPU performance of a SPARC T5–based server with that of a SPARC T4–based server. We find that the SPARC T5 CPU's increased core count, increased I/O throughput, increased memory throughput, and increased clock rate result in 2.7 times the total application throughput of a single SPARC T4 CPU, while latency improves due to reduced contention for CPU access and increased clock rate. The virtualization capabilities of SPARC T5–based and SPARC T4–based servers were used extensively.

After describing the test setup and results, suggested tunings and configurations are presented that help to improve WAS performance, including the use of resource pools, Java Virtual Machine (JVM) tuning options, and Oracle Solaris 11 kernel tuning options.

Background

Previous Testing

Oracle engineers conducted tests in 2013 on WAS v7.0 running the DayTrader 2 Benchmark Application atop Oracle's SPARC T5-2 and SPARC T4-2 servers running Oracle Solaris 10. Their conclusions were as follows:

- The SPARC T5-2 server produces 2.3x the throughput and 11 percent better latency when compared to the SPARC T4-2 server.
- Use of Oracle Solaris Zones virtualization technology improves WAS cluster performance.
- The HTTPS protocol between the client and WAS should be used for increased security; on the SPARC T5 server, this does not result in a performance degradation when compared to the HTTP protocol.

The testing detailed in this paper builds upon (and expands upon) this previous internal work.

DayTrader 3 Benchmark Application

This testing was conducted using workloads from IBM's WebSphere DayTrader 3 benchmark sample.

The WebSphere DayTrader 3 benchmark sample (derived from the original Apache DayTrader 3) provides a suite of workloads for characterizing performance of a Java Platform, Enterprise Edition (Java EE) 6 application server. The workloads consist of an end-to-end web application and a full set of web primitives. Together, the DayTrader 3 application and web primitives provide versatile and portable test cases that are designed to measure aspects of scalability and performance.

The DayTrader 3 benchmark sample is a Java EE 6 application built around an online stock trading system. The application allows users to log in, view their portfolio, look up stock quotes, buy and sell stock shares, and more. Two primary application layers are involved:

- **Presentation Layer**
  The presentation layer consists of several Java servlets and JavaServer Pages (JSPs) that loosely adhere to a model-view-controller (MVC) design pattern. TradeAppServlet is the primary controller servlet responsible for receiving incoming client requests, triggering the desired business logic, and forwarding responses to the appropriate JSP page. Additional servlets and JSPs are used to configure the DayTrader runtime options and manage the supporting database.
- **Business Logic and Persistence Layer**
  The business logic and persistence layer form the bulk of the DayTrader application. The TradeServices interface defines the core set of business operations available in the application, such as register, login, getHoldings, buy, completeOrder, logout, and so on. DayTrader provides three different implementations of these services, corresponding to three commonly used Java EE application design patterns. These implementations are discussed below. Users can switch between these implementations on the configuration page by changed the Runtime Mode.

DayTrader 3 is built on a core set of Java EE 6 web profile technologies and a few Java EE 6 full profile technologies.

Web profile technologies include the following:

- Servlet 3.0
- JSP 2.2
- JavaServer Faces (JSF) 2.0 (new in DayTrader 3)
- Java Persistence API (JPA) 2.0
- Enterprise JavaBeans (EJB) 3.1
- JDBC 4.0

Full profile technologies include the following:

- Java Message Service (JMS) 1.1
- Message-driven beans
- Java API for RESTful Web Services (JAX-RS) 1.1 (new in DayTrader 3)

Two main modes are used for benchmark testing. The mode can be set on the configuration tab of the application, along with many others settings:

- **EJB3 Mode:** Uses EJB beans with JPA to connect to the database
- **Direct (JDBC) Mode:** Uses JDBC to connect directly to the database (Note: This was the mode used for the tests described in this paper.)

Figure 1 provides a high-level overview of the full workload application architecture:

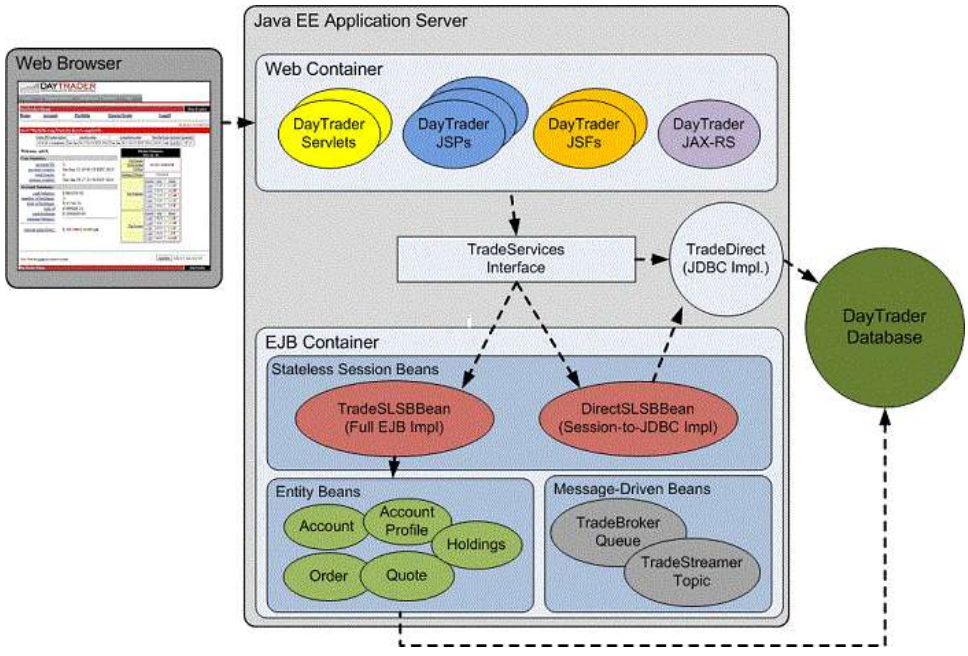

Figure 1: DayTrader 3 Architecture

Test Environment

Overview

DayTrader 3 requires three components: 1) a Java EE application server, 2) a relational database with storage, and 3) a workload generation tool. For this deployment, the following application stack was used:

- Application server: IBM WebSphere Application Server Network Deployment v8.5.5 (Java EE 6–compliant)
- Relational database: IBM DB2 v10.1 (w/FixPack 2) using a small (10 GB) database developed by IBM
- Workload generation: Apache JMeter version 2.9 (Java-based) using scripts developed by IBM

Workload, application, and database layers were implemented in separate virtual systems.

Two simplifying assumptions were made for these tests, in order to minimize the impact of database performance issues:

- A single DB2 instance was used for all tests, running in a SPARC T5 processor–based virtual environment.
- Because the starting database used for these tests was less than 10 GB in size, this common DB2 instance made use of in-memory file systems, rather than traditional physical disk or flash memory. This removed database I/O as a potential performance bottleneck.

Oracle Solaris was used in each virtual test environment; Oracle Solaris 10 was used for the DB2 database layer (Oracle Solaris 11 was not yet supported by DB2 at the time this testing was conducted), while Oracle Solaris 11.1 was used for the application layer environment, as well as the workload generation.

The hardware and virtualization used for these tests included the following:

- Workload generation:
  - Two Sun Fire x4170 M2 servers from Oracle, running Oracle Solaris 11.1
  - No virtualization
- Application layer:
  - One SPARC T5-4 server running a single Oracle VM Server for SPARC (aka LDom)
  - One SPARC T4-4 server running a single LDom
- Database layer: One SPARC T5-4 server running a single LDom using an in-memory file system
- Networking: All environments in this testing were connected via 10 GbE.

It should be noted that the primary goal of this testing was to compare single-CPU performance of the SPARC T5 processor to the SPARC T4 processor. Some additional testing using more than one SPARC T5 processor was done using this testing environment (a SPARC T5-4 server), but it was not a goal of this effort to attempt to maximize performance of WAS v8.5.5 on an entire 4-CPU, 64-core, 512-thread SPARC
T5-4 server.

The test environment, which reflects the recommended deployment for WAS v8.5.5 on a SPARC T5-4 server is shown in Figure 2, and includes the various types of virtualization used for these tests. This deployment is designed to maximize system throughput, minimize latency, and avoid the bottlenecks that are identified in this document. Considerations include the number of virtual CPUs per WebSphere instance, the number of WebSphere instances per LDom, and the number of database instances that can absorb WebSphere transactions.

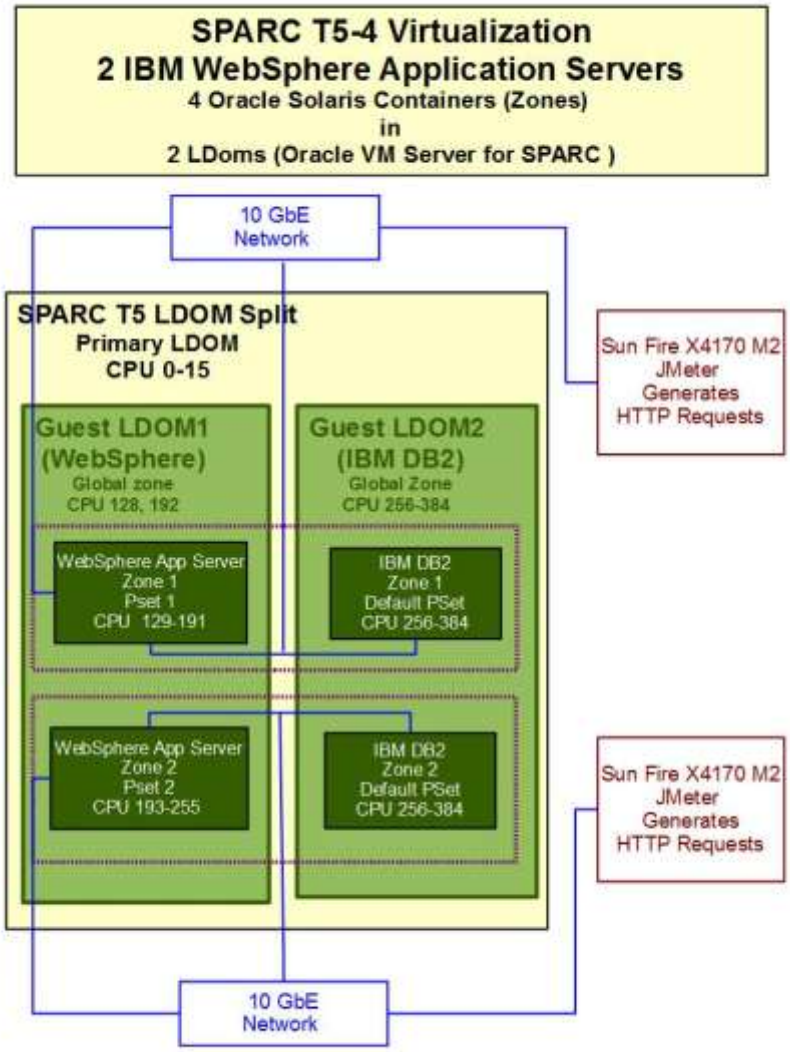

Figure 2: Recommended SPARC T5 Processor–Based Server Topology for WAS

Virtualization Considerations

Deploying a Java EE application server requires many architectural decisions. The decisions made for this particular workload might not work well for every virtualized environment. Further, no attempt was made to compare the following Oracle Solaris Zones deployment to deployments using other virtualization technologies. In summary, a WebSphere deployment into a virtualized environment as described below performs better than a deployment using a single server without virtualization.

The reader should treat the following as guidelines, as their specific WAS environment will likely be different from the one presented here.

Which Virtualization Option?

There are several virtualization options and isolation levels that are available with Oracle Solaris on SPARC (for both Oracle's SPARC T-Series and SPARC Enterprise M-Series servers), including the following:

- Hard partitions using dynamic physical domains (aka PDoms and not applicable to this testing) on SPARC Enterprise M-Series servers.
- Hypervisor-based virtualization such as Oracle VM Server for SPARC (aka LDoms) on SPARC T-Series servers.
- OS virtualization using Oracle Solaris Zones.
- Resource management: Oracle Solaris provides tools for controlling the amount of resources an application or Oracle Solaris Zone receives, such as CPU cycles, physical memory, and network bandwidth.

LDoms and Oracle Solaris Zones, which can be used together, provide the proper level of isolation and flexibility for this test environment. WAS and DB2 were placed in separate guest LDoms on the SPARC T5-4 server, and each was assigned one CPU (16 cores, 128 virtual threads). Whenever possible, physical resources were assigned to minimize the need for a CPU thread to go 'off socket." In the case of physical memory, this meant not assigning memory that was not local to that CPU if it could be helped (thereby avoiding potential Non-Uniform Memory Access [NUMA] issues).

Associating Oracle Solaris Zones with Physical Resources

Within a given LDom several options are available to associate that LDom's resources with particular Oracle Solaris Zones:

- Resource pool association
- Dedicated-CPU resources
- Capped-CPU resources

It was decided that resource pools are the best way for a zone to gain explicit control over the CPU cores in this situation.

To Cluster or Not to Cluster?

WAS clustering allows several WAS instances to coordinate to handle a particular incoming workload. There are several choices available:

- A single WAS instance in one Oracle Solaris Zone
- Multiple WAS instances in one Oracle Solaris Zone
- Multiple-instance WAS cluster across multiple Oracle Solaris Zones

Because of the nature of the workload and database used in these tests, it was determined that the first option (single WAS instance) provided the most straightforward approach, with no demonstrable loss in performance.

Servers per Oracle Solaris Zone

For this application and workload, we obtained best results by using two WAS servers (each within a separate Oracle Solaris Zone) in our WAS LDom. On the database side, two DB2 instances (each within a separate Oracle Solaris Zone) were used. Each WAS instance connected with a corresponding DB2 instance.

Configuration of the SPARC T5-4 Server

Two guest LDoms (in addition to the default primary LDom) were configured using Oracle VM Server for SPARC v3.1 management software for most of the tests. See Appendix A for configuration details.

WAS LDom

- One SPARC T5 processor at 3.6 GHz
- 1 chip x 16 cores x 8 threads = 128 virtual CPUs (vCPUs)
- Memory size: 256 GB
- DayTrader 3.0 for WebSphere Application Server Network Deployment V8.5.5
- OS: Oracle Solaris 11.1.6.4.0
- Zone configuration (two identical zones):
  - Standard `solaris` brand zone
  - `autoboot`: false
  - `ip-type`: shared

Database LDom

- One SPARC T5 processor at 3.6 GHz
- 1 chip x 16 cores x 8 threads = 128 vCPUs
- Memory size: 256 GB
- IBM Enterprise DB2 v10.1.0.2

```
DB21085I  This instance or install (instance name, where
applicable: "db2inst1") uses "64" bits and DB2 code release
"SQL10012" with level identifier "0203010E".
Informational tokens are "DB2 v10.1.0.2", "s121127", "IP23391", and
Fix Pack "2".
Product is installed at "/opt/IBM/db2/V10.1".
```

- OS: Oracle Solaris 10 1/13 s10s_u11wos_24a
- Zone configuration (two identical zones):
  - Standard `native` brand zone
  - `autoboot`: false
  - `ip-type`: shared
  - Memory-based "disks" for DB2 database:
    - Data volume
    - Log volume

Configuration of the SPARC T4-4 Server

One guest LDom (in addition to the default primary LDom) was configured using Oracle VM Server for SPARC v3.1 management software for most of the tests. See Appendix A for configuration details.

WAS LDom

- One SPARC T4 processor at 3.0 GHz
- 1 chip x 8 cores x 8 threads = 64 vCPUs
- Memory size: 128 GB
- DayTrader 3.0 for WebSphere Application Server Network Deployment V8.5.5
- OS: Oracle Solaris 11.1.6.4.0
- Zone configuration (two identical zones):
  - Standard `solaris` brand zone
  - `autoboot`: false
  - `ip-type`: shared

Other Components

JDBC Driver

The following JDBC driver was used for each WAS environment.

```
$ java -cp sqllib/java/db2jcc.jar com.ibm.db2.jcc.DB2Jcc –version
IBM DB2 JDBC Universal Driver Architecture 3.65.77
```

Database Storage

All file systems used by the DB2 instances for this test were created as in-memory RAM disks, to remove DB2 I/O as a potential performance bottleneck. The in-memory `/tmp` file system could also have been used. These RAM disks were created as follows:

- DB2 data files on a 5 GB RAM disk:
  - Create zpool
  - Create ZFS volume:
    - `Recordsize`: 8 KB
    - `Logbias`: throughput
    - `primarycache`: metadata
- DB2 log files on a 5 GB RAM disk:
  - Create zpool: ZFS Intent Log (ZIL) on a separate 2 GB RAM disk
  - Create ZFS volume:
    - `Recordsize`: 128 KB
    - `Logbias`: latency
    - `primarycache`: all

Load Generators

Two identically configured Sun Fire X4170 M2 servers were used to run the Java-based JMeter workload generator for all tests in this document. Each had the following configuration:

- Two Intel Xeon X5675 CPUs at 3.07 GHz
- 2 chips x 6 cores x 2 threads = 24 vCPUs
- Memory size: 96 GB
- OS: Oracle Solaris 11.1
- Apache JMeter Version 2.9

Each JMeter thread implements a "zero think time virtual user." Thus, the workload in this benchmark forms a closed (fixed concurrency) queuing model, which doesn't match a real-world deployment where the load would be formed by a fixed transaction rate.

While this doesn't directly match a real-world scenario, in our experience, this model yields a close approximation to real-world use cases but with a much smaller load generation hardware requirement.

Scaling was tested by increasing the number of JMeter threads until the throughput failed to increase and the response time from WAS quickly increased, indicating that the system under test was constrained by queuing for some limited resource. Final system configuration and tuning recommendations are based on identifying resource limitations (number of database instances) reducing the scarcity of that resource (for example, moving the DB2 transaction log and ZIL to faster storage.)

Tuning for Performance and Scaling

Java Virtual Machine Tuning

The heap size values for this IBM-bundled JVM were varied from 2 GB to 16 GB, and it was determined that 8 GB was the best option. See the "Analysis" section later in this paper for more information.

In addition, the default JDK (version 6) was updated to version 7, as recommended by IBM.

WebSphere Application Server Tuning

The following WAS tunings were suggested by the IBM WAS Team, and were used as starting point; with these parameters, WAS ran well with no bottlenecks observed:

- 200 WebContainer threads:
  a. Click **Servers > Server Types > WebSphere application servers > *server_name* > Thread pools > WebContainer.**
  b. In **General Properties,** set **Minimum Size** and **Maximum size** to 200.
- 220 JDBC pooled connections:
  a. Click **Resources > JDBC > Data Sources > *data_source* > [Additional Properties] Connection pool properties.**
  b. In **General Properties**, set **Minimum connections** and **Maximum connections** to 220.
- Unlimited keep alive:
  a. Click **Servers > Server Types > WebSphere application servers > *server_name***. Then in the Container Settings section, click **Web container > Web container transport chains.**
  b. Select the normal inbound chain for serving requests. This chain is typically called **WCInboundDefault** and listens on port 9080.
  c. Click **HTTP Inbound Channel (HTTP_2)**.
  d. Select **Use persistent (keep-alive) connections** and **Unlimited persistent requests per connection.**

- ResultSets hold when committing a transaction:

  a. Click **Resources > JDBC > Data Sources > *data_source* > [Additional Properties] Custom properties.**

  b. In Preferences, set **resultSetHoldability** to 1.

Oracle Solaris Tuning

IBM recommends that OS large pages be enabled for better WAS performance. The Oracle Solaris 11 kernel has built in support for 2 GB large pages, but Oracle Solaris 10 has no such support by default. Follow this link for configuration steps for Oracle Solaris 10: https://blogs.oracle.com/hardware/entry/bring_2_gb_large_pages.

Oracle Solaris 11 itself is very well tuned "out of the box," and as such, there are very few OS-level tunings required for WAS.

Oracle Solaris Resource Pools

A substantial performance improvement was obtained on the SPARC T5-4 server running a Java EE application server by deploying the instances into Oracle Solaris Zones and binding those zones to cores of the SPARC T5 processor. This is not a surprising result, and it is consistent with other publicly available results. See the "References" section for some examples.

Configuration

- The SPARC T5-4 server has four sockets and 512 hardware threads. We assigned one socket to WebSphere LDOM and one socket to Database LDOM. Thus, each LDOM had 128 virtual processors.

```
# psrinfo -pv
The physical processor has 16 cores and 128 virtual processors (0-
127)
  The core has 8 virtual processors (0-7)
  The core has 8 virtual processors (8-15)
  ...
  The core has 8 virtual processors (120-127)
    SPARC-T5 (chipid 1, clock 3600 MHz)
```

- The "before" test without processor binding:
  Two WebSphere Application Servers were deployed into two Oracle Solaris Zones. The zones shared a 10 GbE port for HTTP traffic and JDBC traffic.

- The "after" test with processor binding:
  Same as "before" test, except vCPUs 1–63 were assigned to Non-Global Zone 1, and processors 65–127 were assigned to Non-Global Zone 2. vCPUs 0 and 64 were left for the global zone.

  Detailed configuration steps can be found in Jeff Taylor's blog:
  https://blogs.oracle.com/taylor22/entry/binding_t4_solaris_containers_to

```
# psrset
user processor set 1: processors 1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
user processor set 2: processors 65 66 67 68 69 70 71 72 73 74 75
76 77
78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116
117 118 119 120 121 122 123 124 125 126 127 128
```

Results

From the tests on the SPARC T5-4 server with and without processor sets, overall throughput and response time was shown to improve approximately 10 percent when the processors were bound, with cross calls and migrations reduced. Details can be found in the "Analysis" section later in this document.

DB2 Tuning

The DB2 instances were architected with the following best practices in mind. In addition

- One common DB2 LDom (running on a SPARC T5-4 server) was used for all tests.
- ZFS volumes were created on RAM disks to remove database I/O bottlenecks from the testing; the transaction log volume should always have a separate ZIL device configured.

The DB2 database was recreated and configured before each test; given the use of RAM disks, this step was extremely quick. It is fully documented in Appendix B.

Within the DB2 instances/zones, very little tuning was required. The recommended tunings made at DB2 installation times are as follows:

```
$ db2osconf -m 256 -n 128
memorySize = 256
maxCPUs = 128
/etc/system setting:

set msgsys:msginfo_msgmni = 20480
set semsys:seminfo_semmni = 24576
set shmsys:shminfo_shmmax = 274877906944
set shmsys:shminfo_shmmni = 24576

Resource control setting for project "group.staff":

projmod -a -K "project.max-shm-memory=(privileged,549755813888,deny)"
group.staff
projmod -a -K "project.max-shm-ids=(privileged,24576,deny)"
group.staff
projmod -a -K "project.max-msg-ids=(privileged,20480,deny)"
group.staff
```

```
projmod -a -K "project.max-sem-ids=(privileged,24576,deny)"
group.staff
```

```
Note: If your project or Zone is bound to a Solaris resource pool, you
need to use -m and -n options of db2osconf to specify the number of
CPUs and the amount of memory use by the resource pool.
```

Analysis

Single-CPU Performance: SPARC T5-4 Server Versus SPARC T4-4

WAS Throughput

Multiple test runs were made, varying the number of simultaneous clients. Figure 3 shows that the single-CPU LDom on the SPARC T5-4 server scales to approximately 2.7x more transactions at saturation point than the single-CPU LDom on the SPARC T4-4 server This is not a wholly surprising result: the SPARC T5 CPU has 16 S3 cores, running at 3.6 GHz, whereas the SPARC T4 CPU has 8 S3 cores, each running at 3.0 GHz.

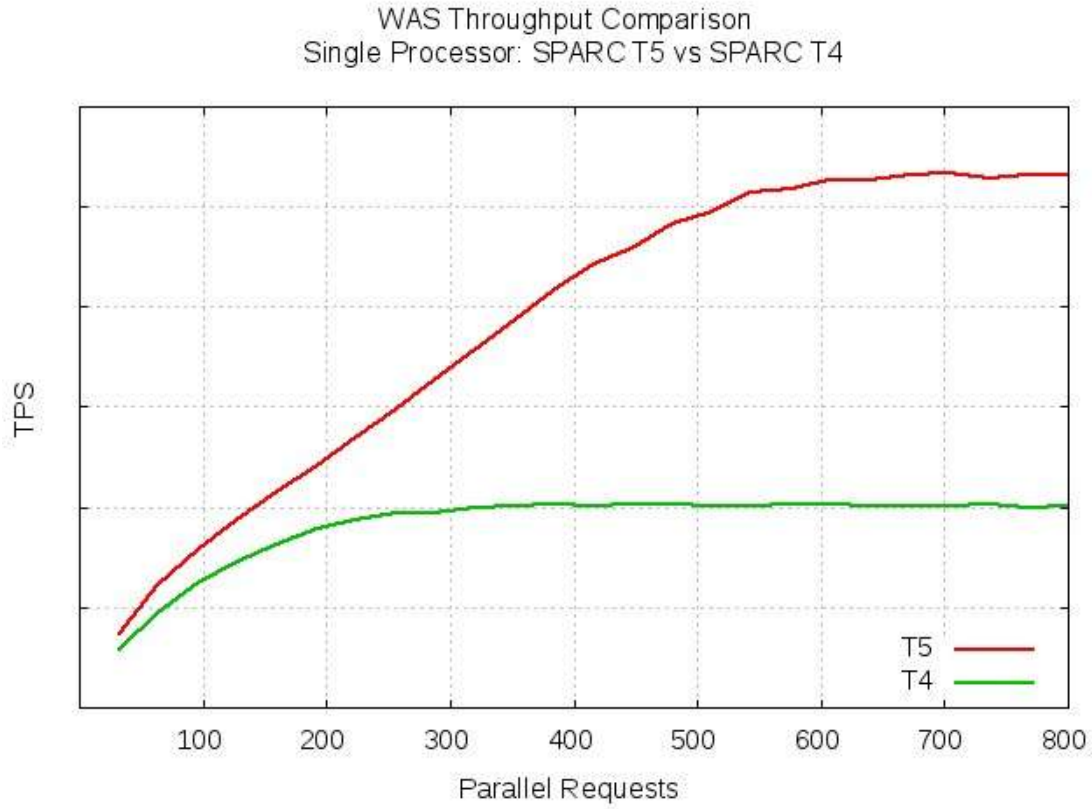

Figure 3: Single-CPU Throughput Comparison

WAS Latency

Again multiple test runs were done, varying the number of simultaneous clients. Figure 4 shows that the single-CPU LDom on the SPARC T5-4 server has better latency characteristics than the single-CPU LDom on the SPARC T4-4 server Under light workload, the SPARC T5 processor has better latency characteristics compared to the SPARC T4 processor due to the higher clock rate; as workload increases, the SPARC T5 processor has better latency characteristics because there is less contention for vCPUs.

With no bottlenecks in the system, these results are unsurprising and demonstrate the impressive and predictable performance/scaling improvements of the newer architecture.

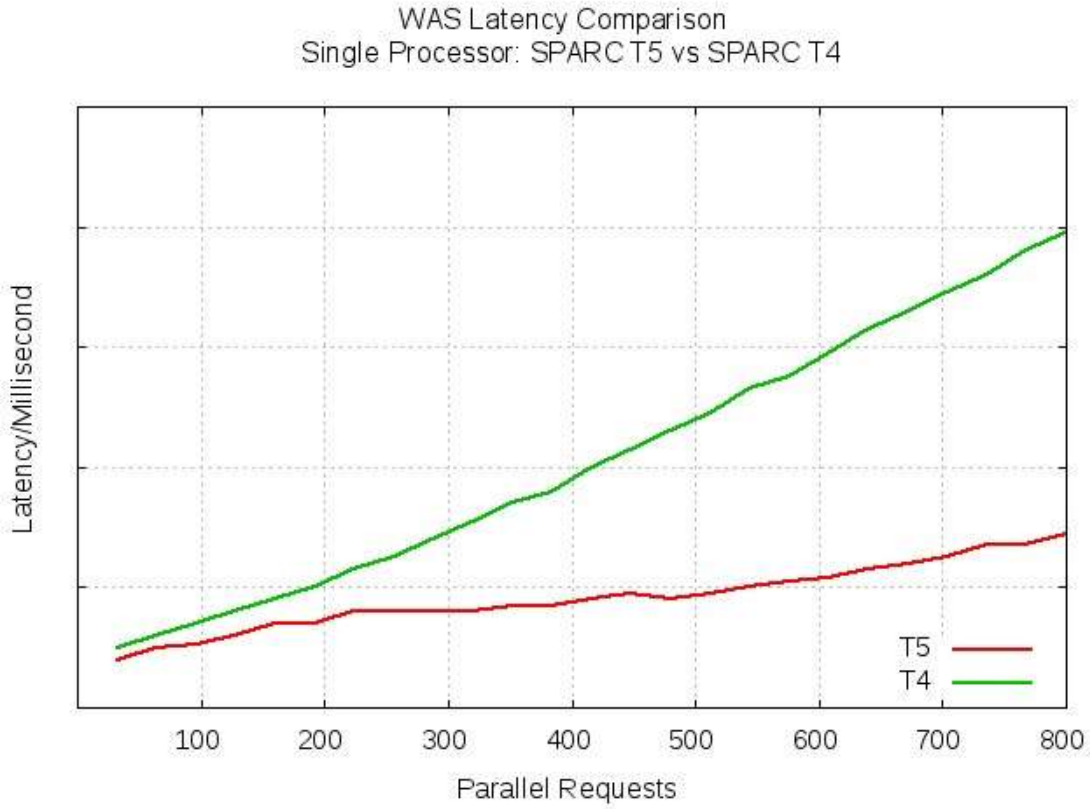

Figure 4: Single-CPU Latency Comparison

WAS CPU Utilization

Over the same range, Figure 5 shows that as workload increases, the SPARC T4 processor becomes saturated much faster than the SPARC T5 processor. The saturation points of throughput and processor utilization are almost identical for the SPARC T4 and SPARC T5 processors. After the saturation point, latency increases greatly, indicating that the processor finally becomes the bottleneck as requests increase.

"CPU utilization" in this case represents the total of System CPU % and User CPU %.

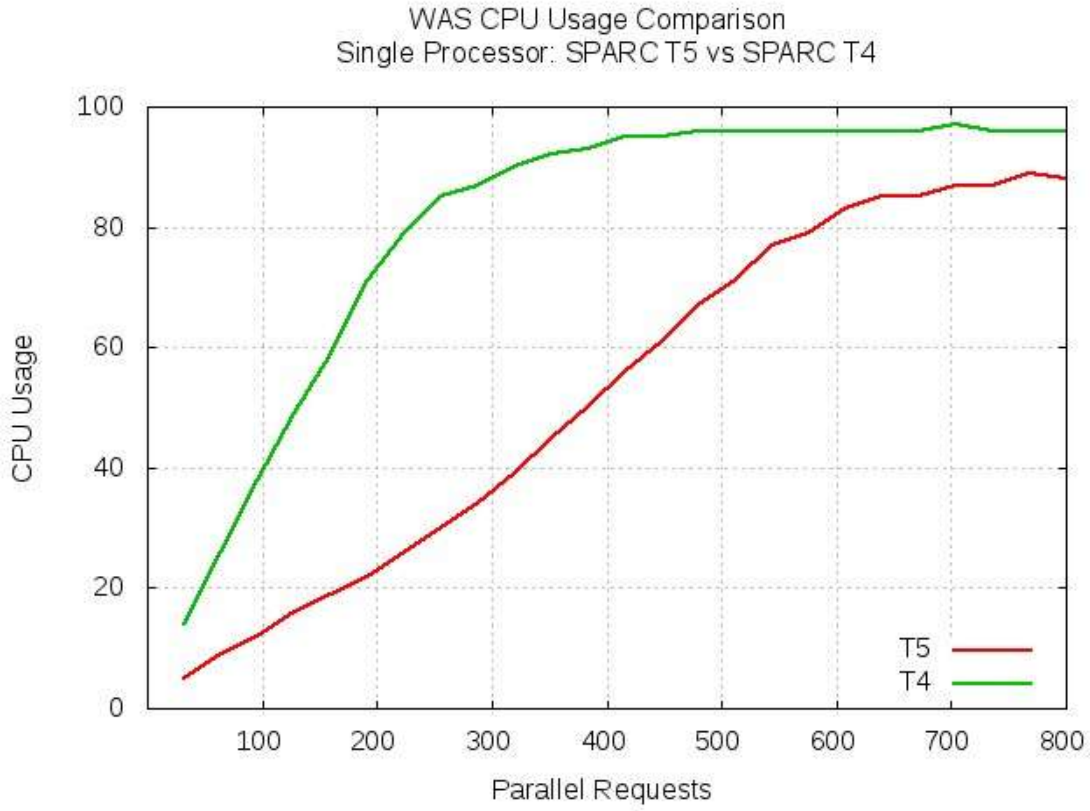

Figure 5: Single-Processor Utilization Comparison

Scalability: One CPU Socket Versus Two on SPARC T5-4

In order to test the scalability of the SPARC T5-4 server, a WAS LDom and a database LDom were added, configured the same as the original WAS LDom and database LDom. Figure 6 shows that the performance is doubled with two sockets, demonstrating linear scalability as the number of vCPUs increases.

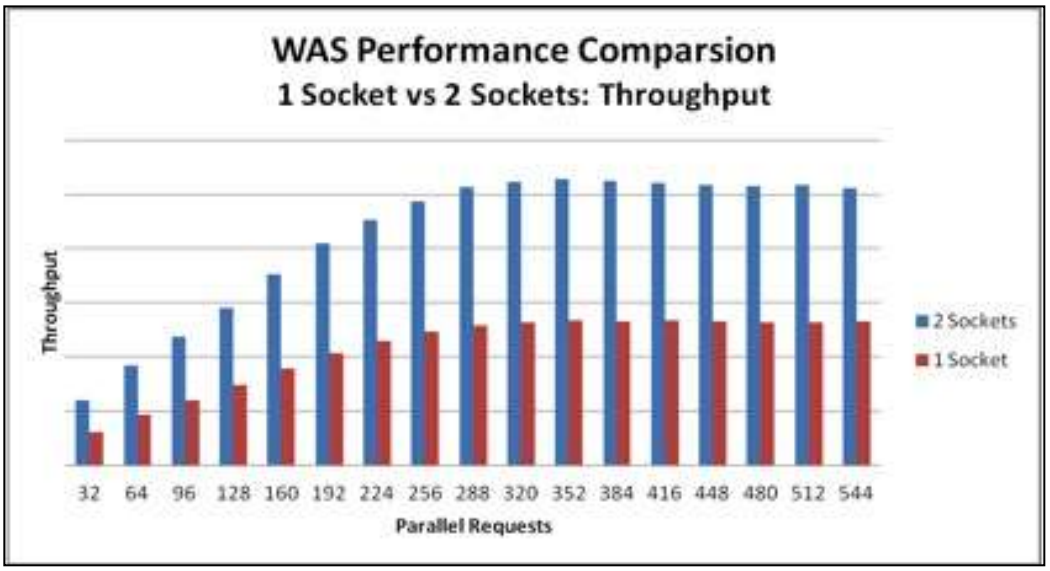

Figure 6: Scalability on SPARC T5-4 Server: One Versus Two Sockets

Scalability: One WAS Instance Versus Two on SPARC T5 CPU

Another oft-asked scaling question centers around varying the number of WAS instances for a given amount of CPU power. For a single SPARC T5 CPU, the performance of one WAS instance in one zone versus two WAS instances in two zones was measured, as shown in Figure 7 and Figure 8.

The best WAS performance was obtained by using two WAS instances per SPARC T5 CPU socket.

The benefit is specific to the system under relatively heavy load. When a light load is applied to the system, the benefit is negligible.

Recommendation: For one SPARC T5 CPU socket, two Oracle Solaris Zones (non-global zones) should be configured, with each zone hosting one WAS instance.

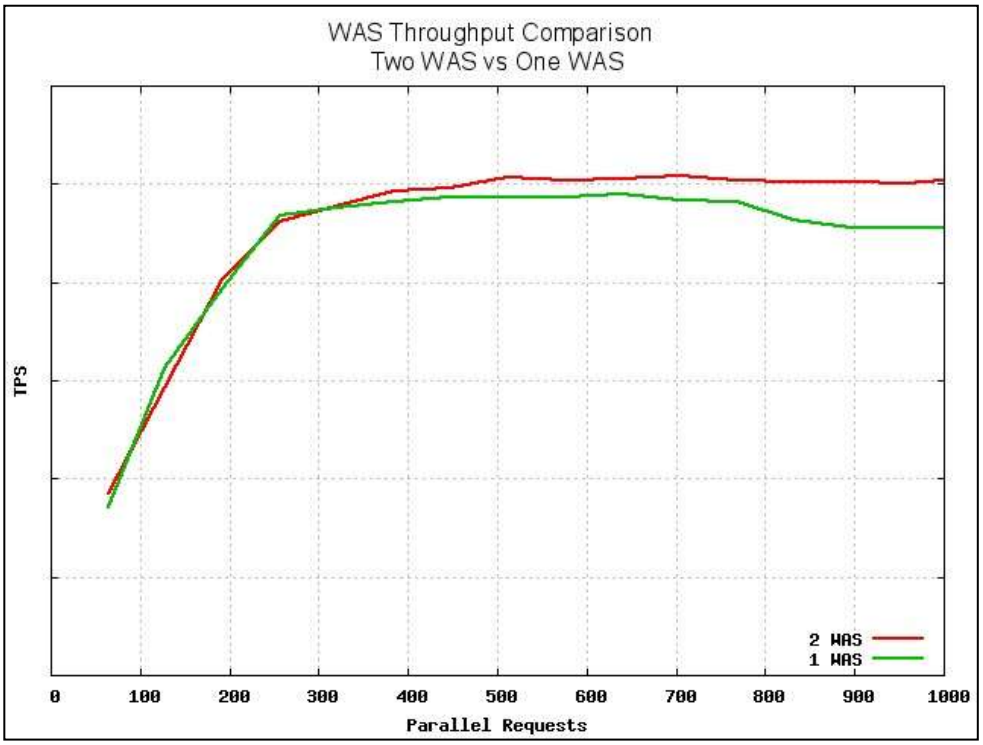

Figure 7: Throughput of One WAS Instance Versus Two WAS Instances

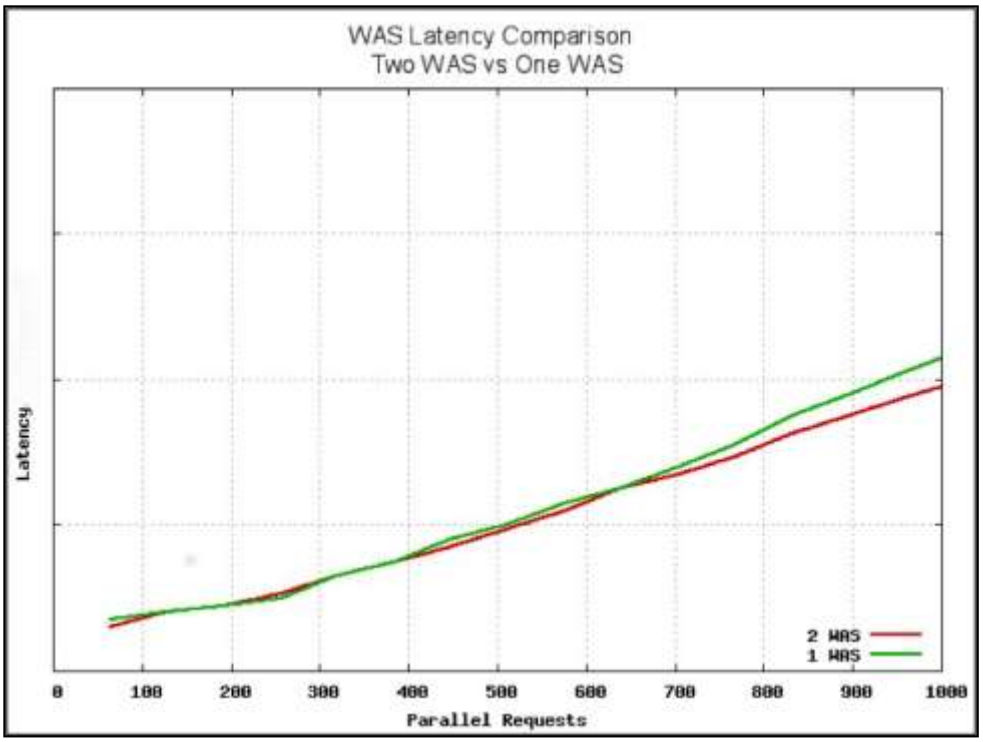

Figure 8: Latency of One WAS Instance Versus Two WAS Instances

Performance and Security Improvements for WAS on SPARC T5

The bulk of the remaining testing and investigation for this project concerned SPARC T5 processor features and performance tuning, rather than straight-up comparisons to previous generations of SPARC CPUs. These tests and investigations helped to form the basis of recommended best practices for running WAS v8.5.5 on SPARC T5–based servers running Oracle Solaris 11.

Impact of SPARC T5 Hardware Encryption for Secure Traffic

SPARC T-Series servers have provided onboard encryption for several generations. Many end users are unaware of this functionality; furthermore, for protocols such as HTTP, hardware encryption can be enabled almost transparently.

SSL

Secure Socket Layer (SSL) is a cryptographic protocol that provides communication security over the internet. WAS supports software SSL to secure access to web resources using the HTTPS protocol, which layers HTTP on top of SSL. Use of software SSL does require precious CPU resources, and an obvious question arose: How does performance of SSL via hardware encryption compare to no encryption or hardware encryption?

By default, WAS starts an HTTPS connector on port 9443 to allow secure access to web applications (in this case, DayTrader 3). For the purpose of this test, the default keystore was used, but this should *not* be done in production environments. Since the same keystore is distributed on every instance of WAS, the key is no longer secret. For configuring the solution to use SSL in a production environment, please refer to the link to "Using SSL in Production Environments" in the "References" section at the end of this paper.

KSSL

Kernel SSL proxy (KSSL) is an Oracle Solaris kernel module that acts as a server-side SSL protocol, and can be used for offloading such operations as SSL/TLS-based communication, SSL/TLS termination, and reverse proxying for end user applications. It provides processing of SSL traffic in the kernel and, thus, improves performance by avoiding context switches and directly accessing the kernel providers of the Oracle Solaris Cryptographic Framework.

KSSL is configured in the kernel and receives clear-text data from and sends clear-text data to the application. From the client side, the application is an SSL server; the application side is unaware of any SSL, and the incoming and outgoing traffic are all clear text.

In Oracle Solaris, hardware encryption (if available) can be specified as KSSL.

For the tests performed for this white paper, the DayTrader application was configured to be unaware that the requests sent from JMeter were using SSL. KSSL received SSL traffic on port 7443, performed processing, and passed clear-text data to DayTrader 3, listening on port 9080. Similarly, for the outgoing traffic, DayTrader 3 sent clear-text data and KSSL produced SSL encryption and sent data back to JMeter. Throughout this process, DayTrader 3 was unaware of any traffic encryption or the need to set up SSL.

For more information, see the link to the Oracle documentation in the "References" section at the end of this paper.

Results

Tests were run with no changes to either DayTrader 3 or WAS to compare performance as the number of parallel clients increase. The following configuration was used:

- No traffic encryption (HTTP)
- Software traffic encryption (HTTPS—software SSL)
- Hardware traffic encryption (HTTPS—hardware SSL via KSSL)

From Figure 9, Figure 10, and Figure 11, the following is observed:

- WAS using hardware encryption (via KSSL) performs much better than software SSL encryption and almost as well as no encryption, both in throughput and latency.

- The most computationally expensive part of an SSL session is the SSL handshake, but with KSSL, all SSL operations, including the SSL handshake and session state, are performed asynchronously in the Oracle Solaris kernel. The kernel automatically uses the Oracle Solaris Cryptographic Framework for offloading operations to the underlying hardware cryptographic accelerators without any extra effort.
- It is, therefore, unnecessary to sacrifice performance for more secure web-tier traffic on SPARC T5–based servers.

These tests showed the benefits of Oracle Solaris KSSL and SPARC T5 hardware encryption for one very simple use case. Applications with larger amounts of client/server traffic might demonstrate even greater performance improvements using this approach.

Recommendation: Make use of KSSL and hardware encryption for HTTP traffic to and from WAS whenever possible on SPARC T5–based servers; the performance penalty is minor, and the security advantage is significant.

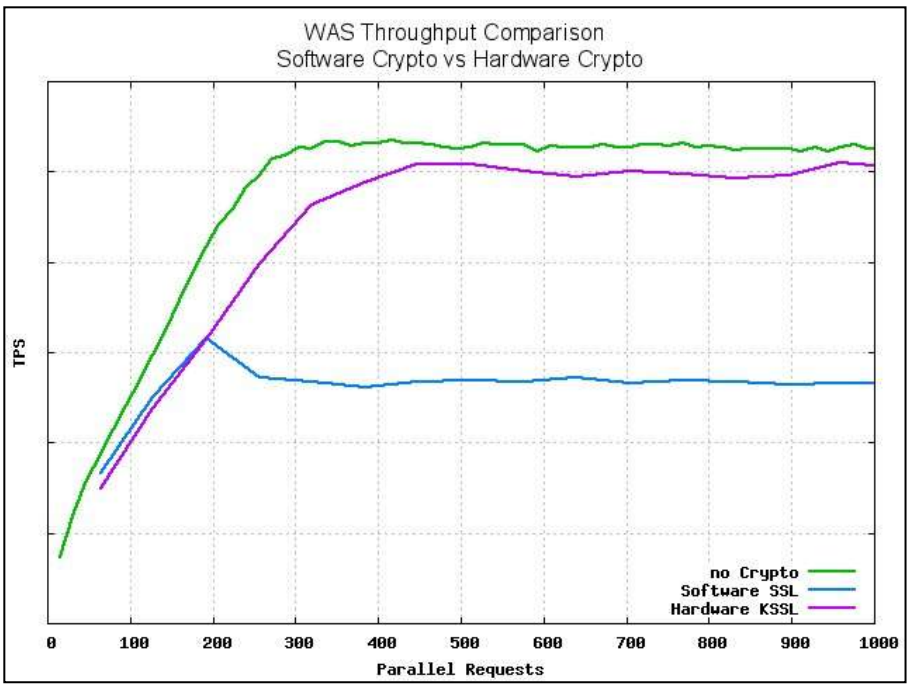

Figure 9: Throughput Comparison of Cryptographic Options

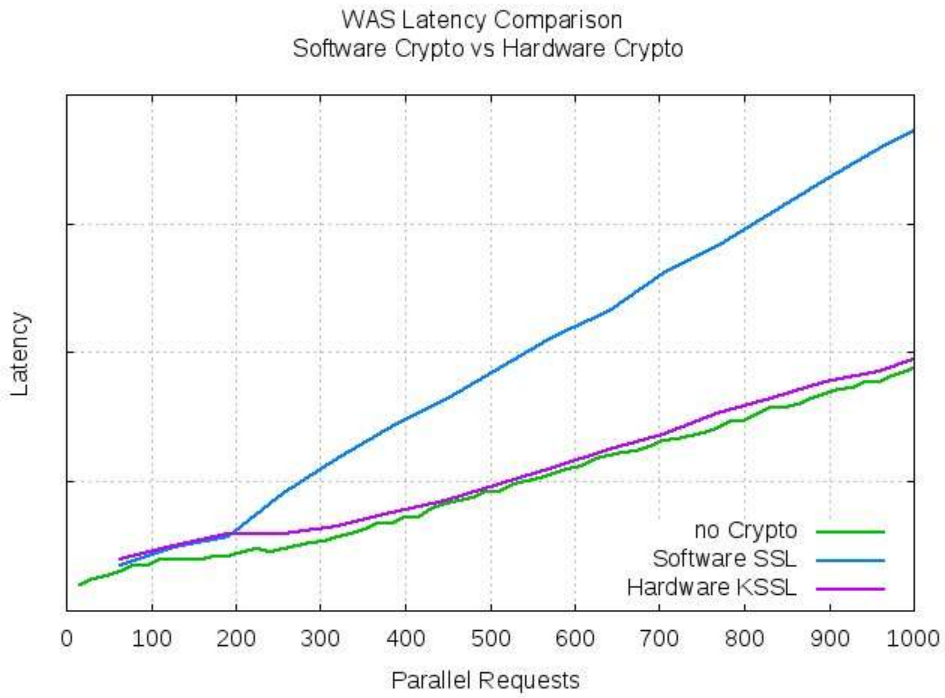

Figure 10: Latency Comparison of Cryptographic Options

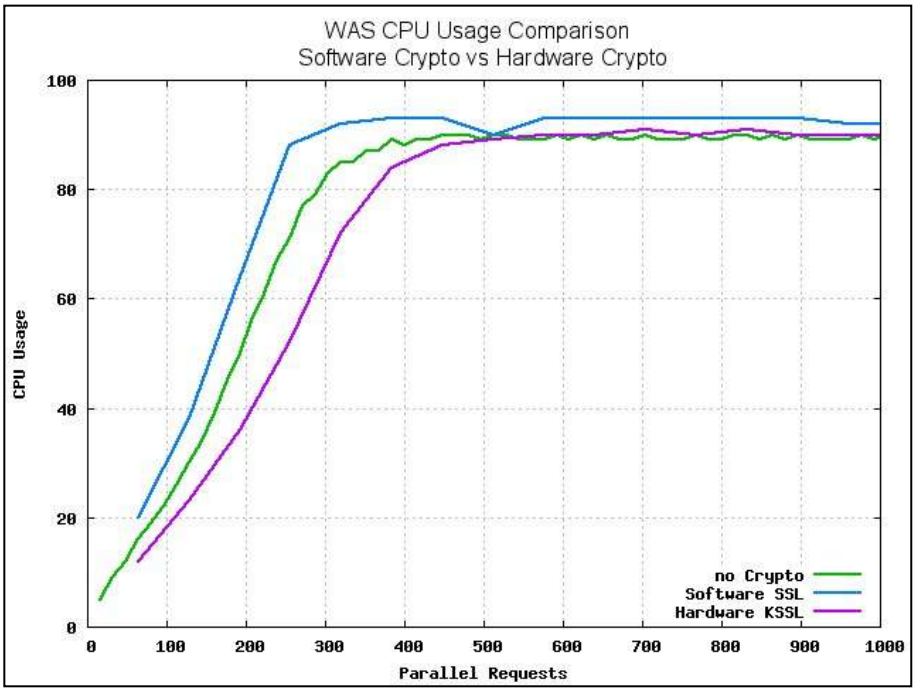

Figure 11: CPU Usage Comparison of Cryptographic Options

Processor Set Binding

As the "Tuning for Performance and Scaling" section described, WAS instances were deployed into Oracle Solaris Zones, each bound to a 63-vCPU processor set (two vCPUs were reserved for the global zone).

Figure 12 shows that the SPARC T5-4 server scales with as much as 5 percent higher maximum throughput when each of the WAS instances is bound to an Oracle Solaris processor set. The benefit is specific to a system under relatively heavy load. When a lighter load is applied to the system, the benefit is negligible.

Processor sets did not appear to impact application latency significantly (see Figure 13).

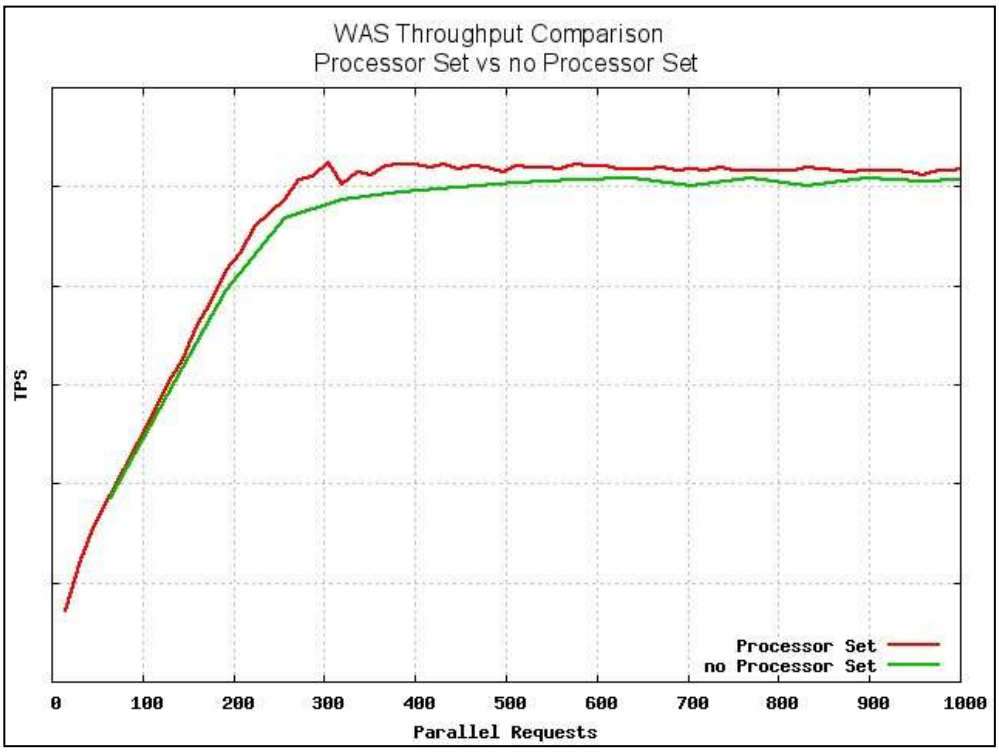

Figure 12: Impact of Processor Sets on WAS Zones—Throughput

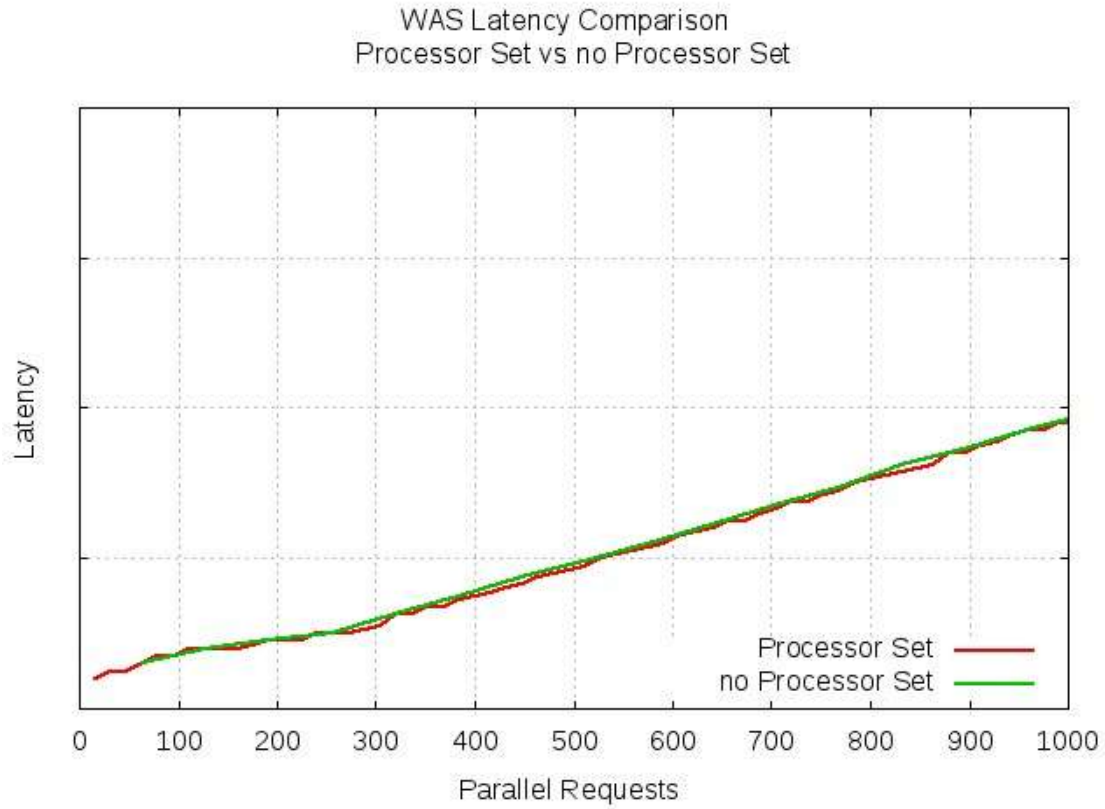

Figure 13: Impact of Processor Sets on WAS Zones—Latency

Varying JVM Heap Size for WAS

The JVM heap is where the objects of a Java program live. It is a repository for live objects, dead objects, and free memory. When an object can no longer be reached from any pointer in the running program, it is considered "garbage" and ready for collection. A best practice is to tune the time spent doing garbage collection to within 5 percent of execution time.

The JVM heap size determines how often and how long the JVM spends collecting garbage. If the heap size is too large, full garbage collection is slower but occurs less frequently. If the heap size is too small, full garbage collection is faster but occurs more frequently. For more information, consult the "References" section at the end of this paper.

In order to tune the heap size to minimize the time the JVM spends doing garbage collection while maximizing the performance of WAS at any given time, JVM heap sizes of 2 GB, 3 GB, 4 GB, 8 GB, and 16 GB were used, and performance was monitored. Again, DayTrader 3 is an example of a small application, and many real-world applications are much larger. The results seen here are nonetheless instructive.

Figure 14 and Figure 15 show that in this test scenario, an 8 GB JVM heap size gives the best results; it yields better throughput and latency than the smaller heap size values. Using larger values (such as 16 GB) does not provide incremental benefits. Figure 16 shows that when the heap size is more than 8 GB, the time WAS spends on garbage collection is below 5 percent.

Recommendation: For WAS on SPARC T5–based servers, an 8 GB JVM heap size is the smallest value that provides optimal throughput and latency.

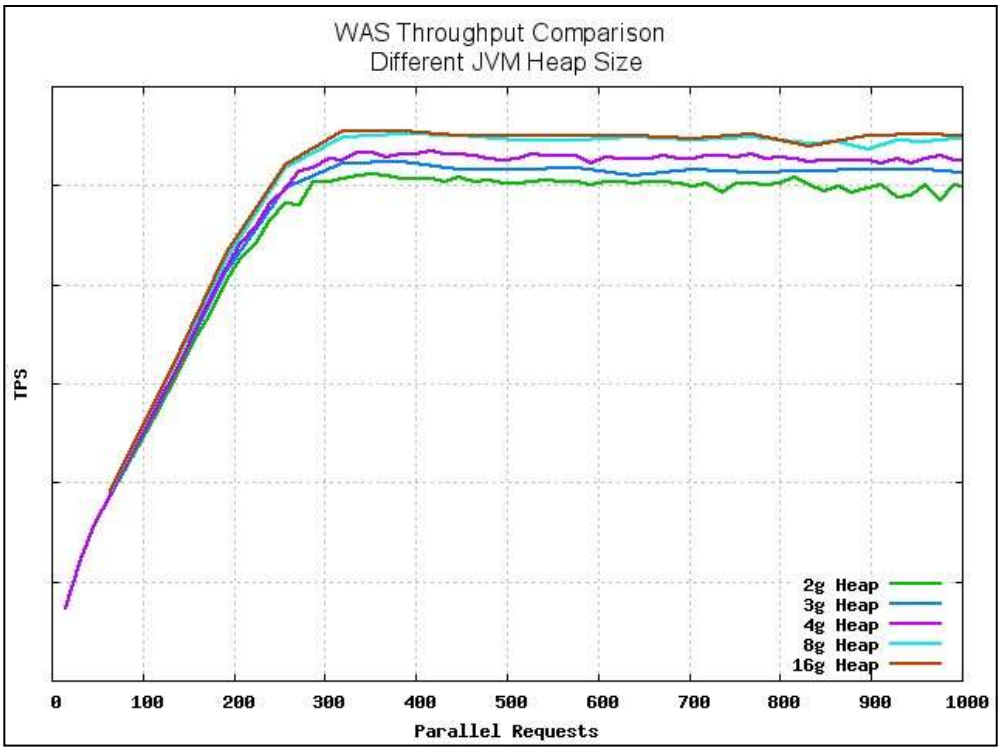

Figure 14: Impact of JVM Heap Sizes on Throughput

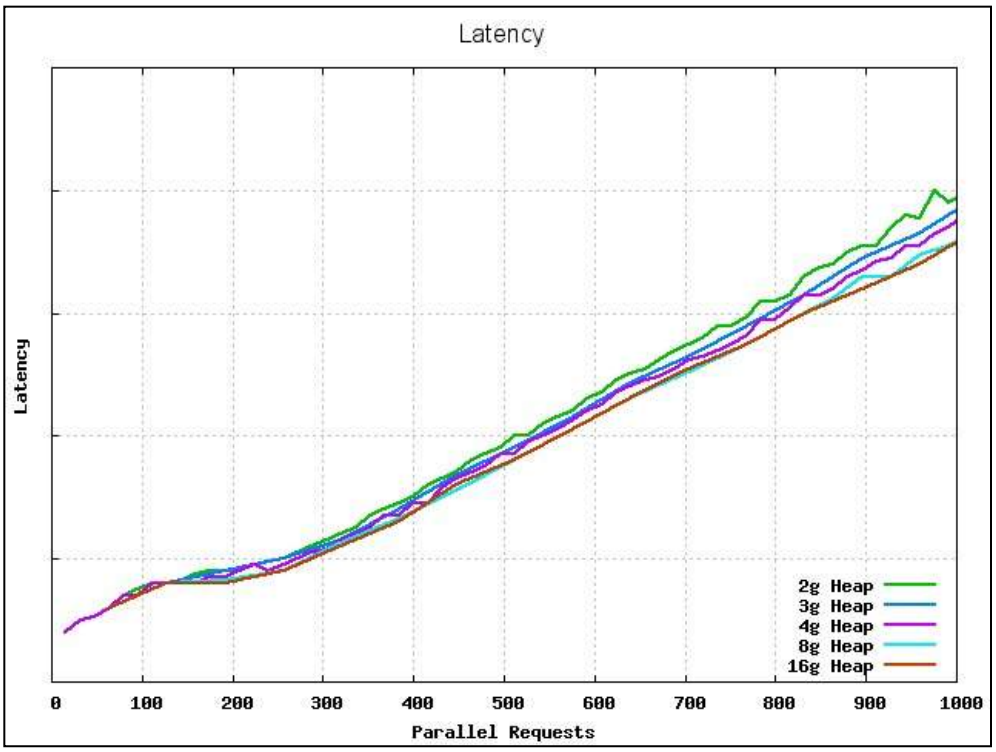

Figure 15: Impact of JVM Heap Size on Latency

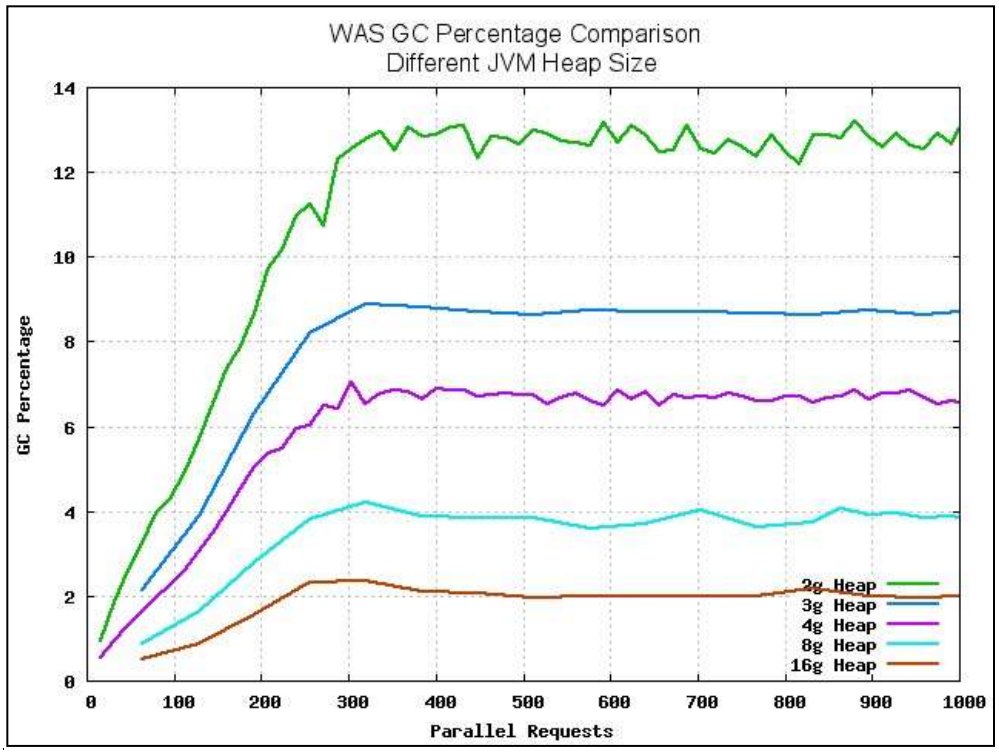

Figure 16: Impact of JVM Heap Size on Garbage Collection

Interrupt Fencing

By default, processors must service a myriad of interrupts from throughout the system and, traditionally, this is done across all available hardware processing threads. *Interrupt fencing* is a technique that allows those interrupts to be serviced only by designated processor threads, thereby providing potential performance improvements to the application itself on the remaining threads.

As described in the "Tuning for Performance and Scaling" section, the two WAS instances were deployed into Oracle Solaris Zones using the following configuration:

- The first WAS zone was bound to vCPUs 1–63 (relative to the LDom, not to the entire server)
- The second WAS zone was bound to vCPUs 65–127
- vCPUs 0 and 63 were reserved for the global zone

In this configuration, performance of the second WAS instance is always about 15 percent better than the performance of the first WAS instance. After carefully observing the CPU statistics, numerous interrupts were noted on the first half of the vCPUs (that is, Processor Set 1), which are bound to the first WAS zone. Further investigation showed that this activity in Processor Set 1 was due to an influx of interrupts from external devices—in this case, network cards and, to a lesser extent, disk operations. Traffic to and from the load generators causes interrupts in order to service the network cards, and this temporarily suspends the execution flow of the thread running on the interrupted processor. All of this impacts the performance of the first WAS instance.

In order to strike a better balance between the interrupts and performance (measured in transactions per second [TPS]) of the two WAS instances, interrupts were disabled on vCPUs 4–63 and 68–127, so that vCPUs 0–3 and 64–67 focused on dealing with any and all the interrupts for this LDom.

Figure 17 shows distribution of interrupts across the 128 vCPUs of the LDom running WAS instances. The green line shows the distribution without interrupt fencing, and most of the interrupts are handled by vCPUs in processor set 1, competing with the WAS instance running in that processor set. The red line shows the distribution when interrupts are "fenced out" from most of the vCPUs in the processor sets, and are handled by six vCPUs: two in the global domain and two from each of the processor sets. These six vCPUs handle the spike in interrupts, while the remaining interrupts are more evenly distributed. With interrupt fencing, the WAS instance in processor set 1 runs without competing with the interrupts for compute power.

Figure 18 shows that, without interrupt fencing (green bar), the second WAS instance's throughput is much higher than the first WAS instance's throughput; once fenced (red bar), the difference between the throughput of the two instances decreases significantly, while the overall throughput remained the same.

Recommendation: If there are multiple WAS instances running on one SPARC server, they should be deployed on separate Oracle Solaris Zones. vCPUs should be bound equally across zones. If balanced throughput across multiple WAS instances is desired, interrupts should be fenced elsewhere.

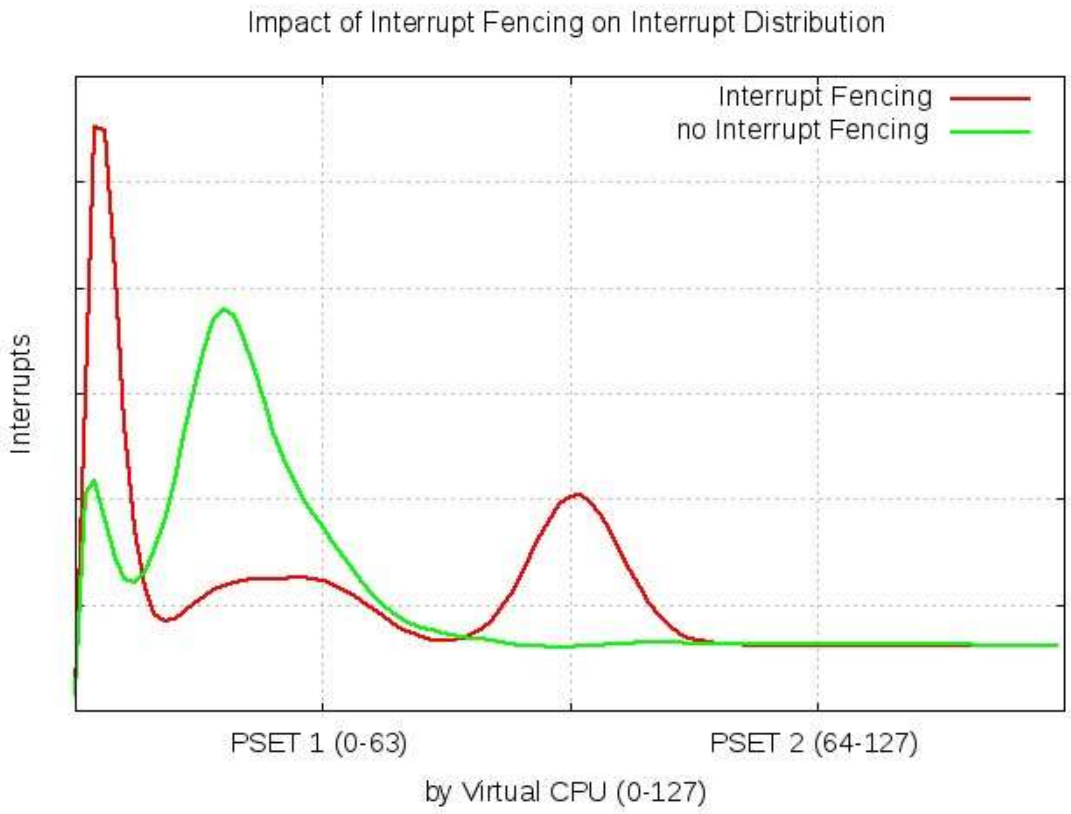

Figure 17: Distribution of Interrupts Across the Virtual CPUs

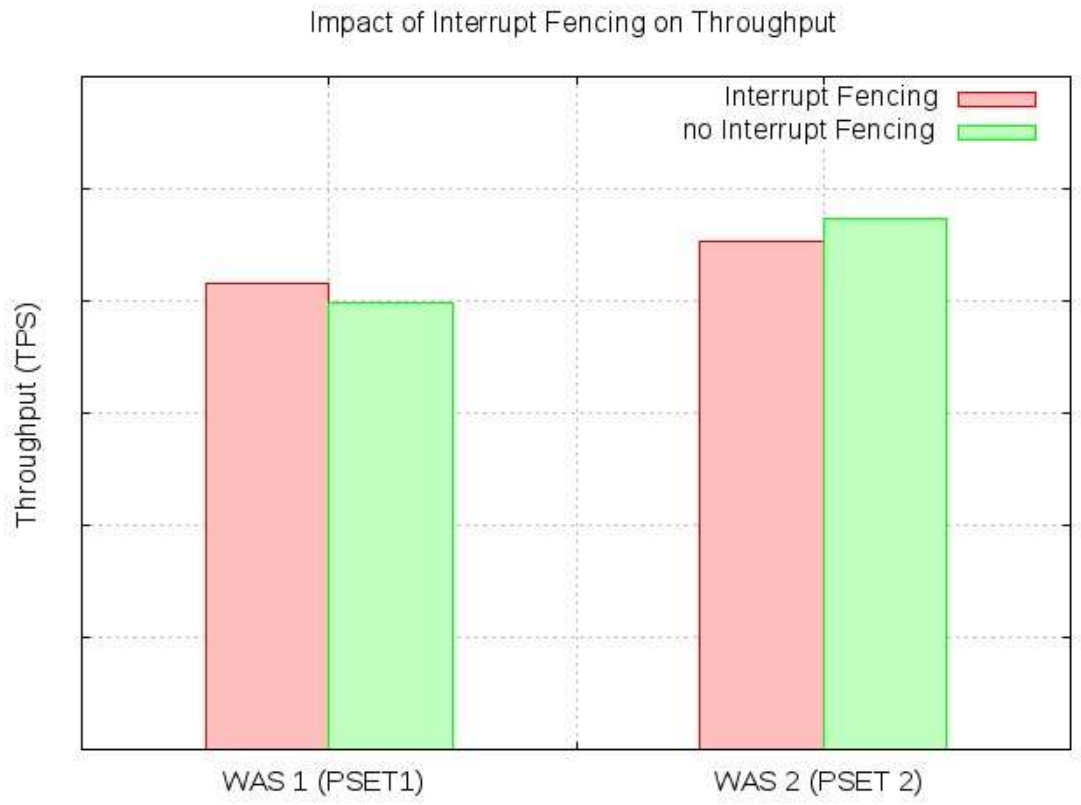

Figure 18: Throughput Changes for the Fenced Versus Unfenced Tests

Network Interface Cards as a Possible Bottleneck

Network traffic for these tests was never expected to overload the single 10 GbE card provided to each load generation system, WAS instance, and DB2 instance. Nonetheless, as the number of parallel requests increased, and with it network traffic between each tier, it was suggested that a comparison be made between two otherwise-identical configurations:

- One 10 GbE network interface card (NIC) serving each WAS instance (HTTP traffic to/from the load generator and JDBC traffic to/from DB2 intermingled)
- Two 10 GbE NICs serving each WAS instance (separate NIC for HTTP and JDBC traffic)

The intent was to show whether a second NIC per WAS instance would improve performance, while at the same time generating more interrupt events for the CPUs.

Figure 19 shows the interrupt distribution of both configurations; with two NICs per WAS instance, the interrupts were distributed more evenly, and the performance of both WAS instances was similar.

Figure 20 shows that throughput was slightly better when all HTTP and JDBC traffic shared one 10 GbE NIC, and slightly worse when two separate 10 GbE NICs were used. While seemingly counterintuitive, this might be explained by the fact that the high rate of interrupts can disrupt thread execution enough to raise performance issues, not only because of the WAS instance running threads being interrupted but also because of the cache effect of the processor running interrupted WAS threads. The text and data for the interrupt threads can displace data in the hardware cache lines that were part of the WAS threads' address spaces, causing cache misses when the interrupted WAS threads resume execution. With one NIC, network interrupts are restricted to one processor set, which results in better performance than two NICs with network interrupts distributed to two processor sets.

Figure 21 shows that WAS latency is better (lower) for one 10 GbE NIC than for two 10 GbE NICs, in accord with the throughput results from Figure 20.

Recommendation: For WAS running on a SPARC T5–based server, use a single NIC unless higher security is needed to separate the HTTP and JDBC traffic. System administrators should monitor network utilization to ensure that network bandwidth does not become a system bottleneck, thereby impacting system response time and performance.

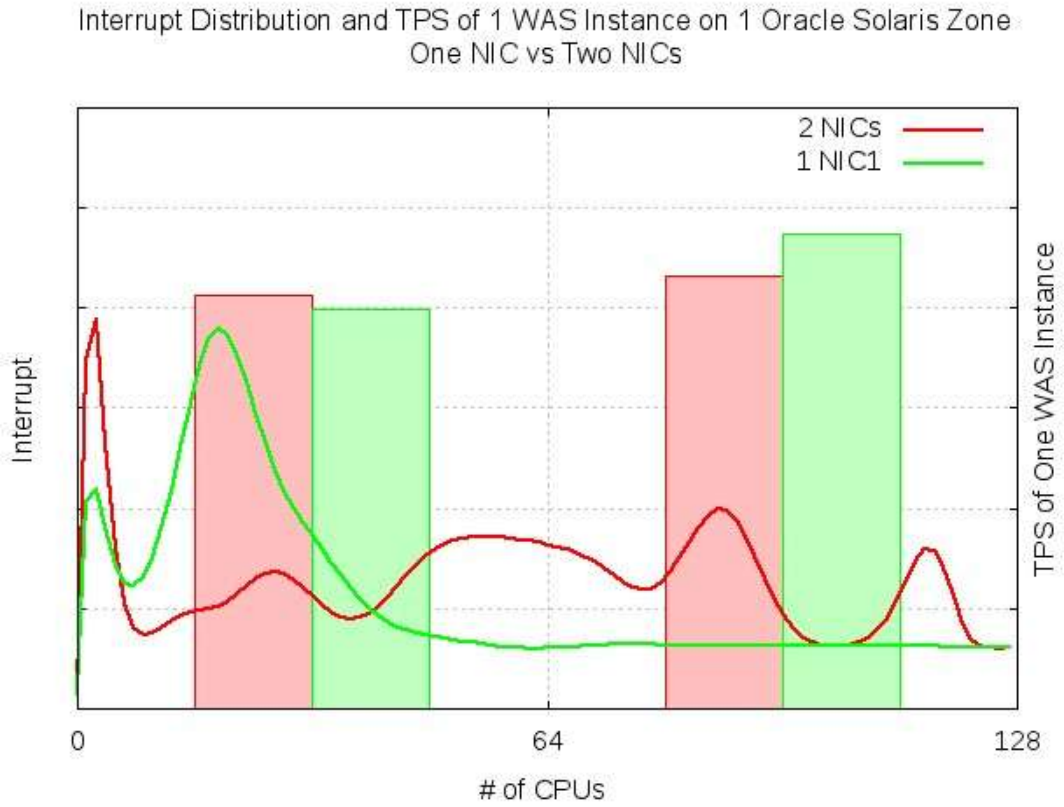

Figure 19: One Versus Two NICs per WAS Instance: Interrupt Distribution

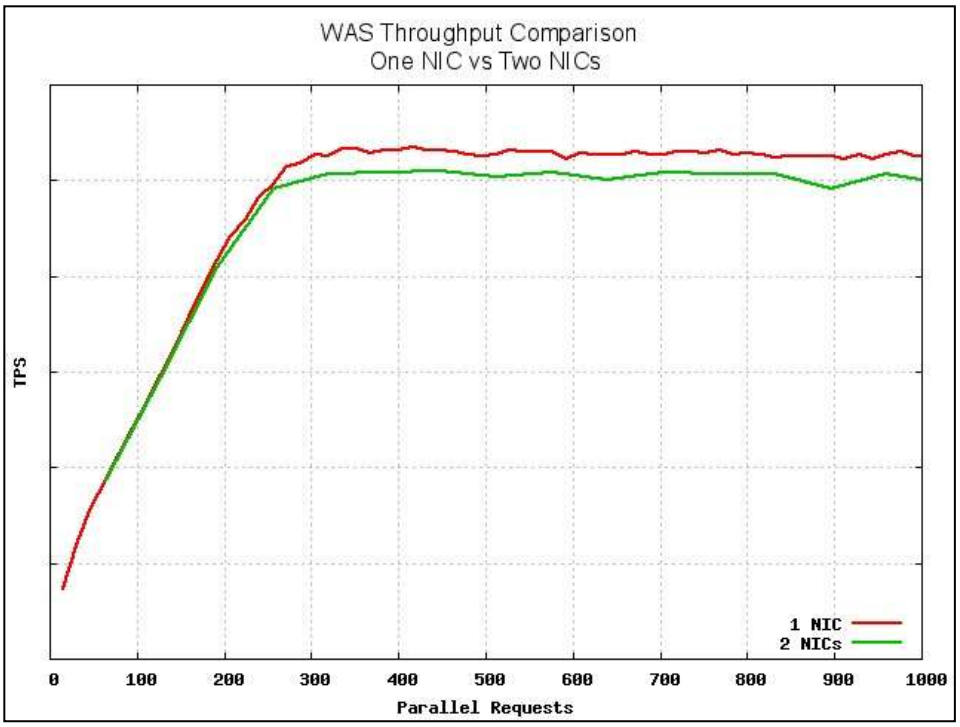

Figure 20: One Versus Two NICs per WAS Instance: Throughput

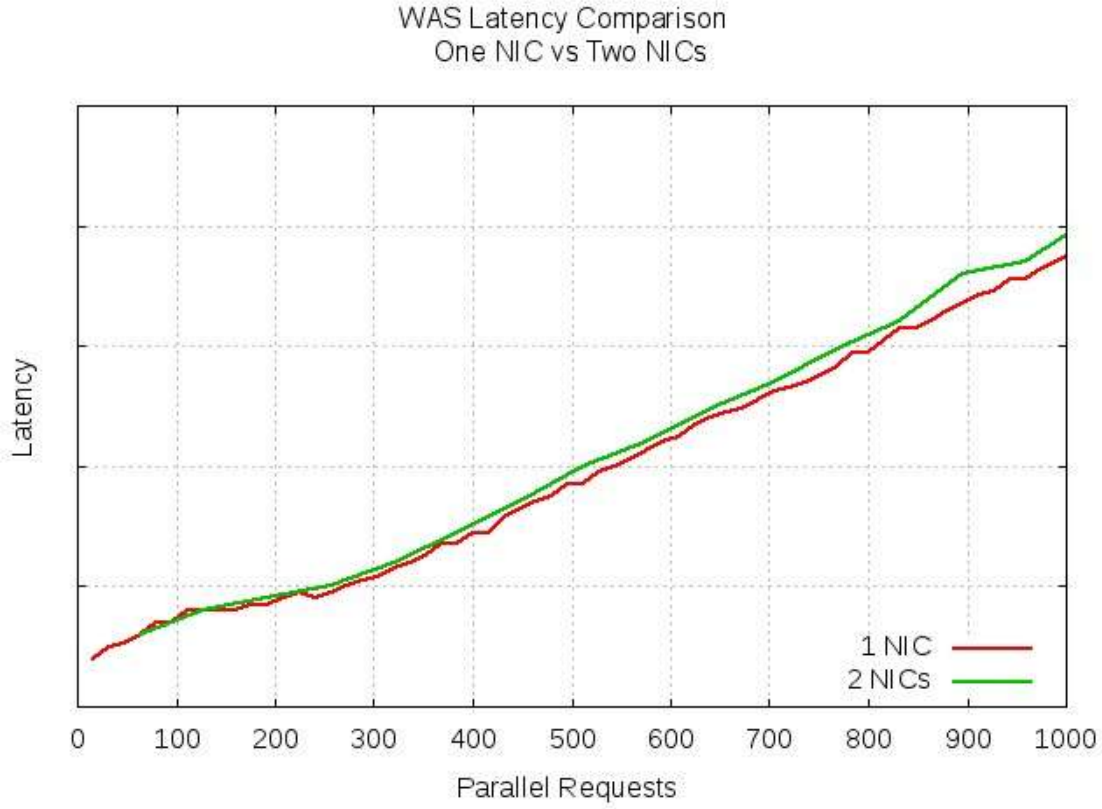

Figure 21: One Versus Two NICs per WAS Instance: Latency

Conclusion

From the testing conducted for this paper, the following conclusions were drawn:

- SPARC T-Series servers provide a scalable, dependable, and performant platform for the deployment of WAS instances.
- Application architects should have confidence in deploying WAS to multiple "building blocks" consisting of one or more SPARC CPUs; these building blocks can be instantiated as physical or virtualized resources.
- On a CPU-to-CPU comparison basis, WAS v8.5.5 demonstrates 2.7x better throughput on SPARC T5 CPUs than on SPARC T4 CPUs, with improved response time.
- Use of Oracle VM Server for SPARC and Oracle Solaris Zones virtualization technologies not only allow clean separation of virtual environments, but also allow for full and efficient use of physical resources, yielding maximum performance.
- SPARC T5–based servers demonstrate linear scalability for WAS as the number of CPU cores increases.
- Careful physical resource management via Oracle Solaris, when done in concert with the above-mentioned virtualization technologies, can improve performance for WAS instances on SPARC-based servers running Oracle Solaris.
- The use of the on-chip encryption capabilities of SPARC T5 processor provides a straightforward and powerful way to obtain increased security for WAS-based systems, with minimal performance impact.

Acknowledgements

This paper is the result of many person-months of effort from both Oracle and IBM. On the Oracle side, the authors wish to thank their leadership team (especially Tom Gould) for their guidance and support and the Compute Resources Labs Team (Les Leong, Stephen Dill, and Kevin Thai) for their professional work in the installation, configuration, and support of the hardware and software infrastructure used in this project. Thanks as well to those who proofread and provided invaluable feedback on this paper. A special thanks to Jeff Taylor of the Oracle Platform Integration group who, along with Niting, conducted the previous set of tests of IBM WebSphere Application Server and Oracle's SPARC servers; many of the themes for this paper echo the thoughtful and thorough work he presented within Oracle.

References

- Oracle's SPARC processor public roadmap: http://www.oracle.com/us/products/servers-storage/servers/sparc/oracle-sparc/sparc-roadmap-slide-2076743.pdf
- DayTrader 3 benchmark sample: https://www.ibmdw.net/wasdev/docs/measuring-performance-daytrader-3-benchmark-sample/
- DayTrader 3 download: http://www-01.ibm.com/software/webservers/appserv/was/performance.html
- IBM Redbook: *WebSphere Application Server V8.5 Administration and Configuration Guide for the Full Profile*, Albertoni, et al: http://www.redbooks.ibm.com/redbooks/pdfs/sg248056.pdf
- *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*: http://docs.oracle.com/cd/E23824_01/html/821-1460/index.html
- "LDoms with Solaris 11" from Orgad Kimchi's "The art of virtualization" blog: https://blogs.oracle.com/vreality/entry/ldoms_with_solaris_11
- "Java EE Application Servers, SPARC T4, Solaris Containers, and Resource Pools" from Jeff Taylor's blog: https://blogs.oracle.com/taylor22/entry/binding_t4_solaris_containers_to
- Using SSL in production environments: http://pic.dhe.ibm.com/infocenter/prodconn/v1r0m0/index.jsp?topic=%2Fcom.ibm.scenarios.wmqwassecure.doc%2Ftopics%2Fcfgssl_was.htm
- Tuning JVM heap sizes: http://docs.oracle.com/cd/E15523_01/web.1111/e13814/jvm_tuning.htm#i1141344
- KSSL defined: http://docs.oracle.com/cd/E23824_01/html/821-1474/kssl-5.html
- "Creating a ZFS Storage Pool With Log Devices" in *Oracle Solaris Administration: ZFS File Systems*: http://docs.oracle.com/cd/E23824_01/html/821-1448/gaypw.html#gffyt

Appendix A: Details for Oracle VM Server for SPARC

SPARC T5-4 Server Configuration

```
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    16    64768M   0.2%  0.2%  12d 12h 36m

UUID
    5f3eb1f6-1ee0-e932-adef-87122ddf27e3

MAC
    00:10:e0:3a:d1:1c

HOSTID
    0x863ad11c

CONTROL
    failure-policy=ignore
    extended-mapin-space=on
    cpu-arch=native
    rc-add-policy=
    shutdown-group=0

DEPENDENCY
    master=

CORE
    CID    CPUSET
    0      (0, 1, 2, 3, 4, 5, 6, 7)
    1      (8, 9, 10, 11, 12, 13, 14, 15)

VCPU
    VID    PID    CID    UTIL NORM STRAND
    0      0      0      0.3% 0.3%   100%
    1      1      0      0.1% 0.1%   100%
    ...
    14     14     1      0.1% 0.1%   100%
    15     15     1      0.1% 0.1%   100%

MEMORY
    RA               PA               SIZE
    0x30000000       0x30000000       64768M

CONSTRAINT
    threading=max-throughput
    physical-bindings=memory

VARIABLES
    boot-
device=/pci@300/pci@1/pci@0/pci@4/pci@0/pci@c/scsi@0/disk@w5001517bb2a7d34b,0:a disk
net
    pm_boot_policy=disabled=0;ttfc=2000;ttmr=0;

IO
    DEVICE                              PSEUDONYM        OPTIONS
    pci@300                             pci_0
    pci@340                             pci_1
    pci@380                             pci_2
    pci@3c0                             pci_3
    pci@400                             pci_4
    pci@440                             pci_5
    pci@480                             pci_6
    pci@4c0                             pci_7
    pci@300/pci@1/pci@0/pci@6           /SYS/RCSA/PCIE1
    pci@300/pci@1/pci@0/pci@c           /SYS/RCSA/PCIE2
    pci@300/pci@1/pci@0/pci@4/pci@0/pci@c /SYS/MB/SASHBA0
    pci@300/pci@1/pci@0/pci@4/pci@0/pci@8 /SYS/RIO/NET0
    pci@340/pci@1/pci@0/pci@c           /SYS/RCSA/PCIE4
    pci@380/pci@1/pci@0/pci@a           /SYS/RCSA/PCIE9
    pci@380/pci@1/pci@0/pci@4           /SYS/RCSA/PCIE10
    pci@3c0/pci@1/pci@0/pci@8           /SYS/RCSA/PCIE12
    pci@400/pci@1/pci@0/pci@e           /SYS/RCSA/PCIE5
    pci@400/pci@1/pci@0/pci@8           /SYS/RCSA/PCIE6
    pci@440/pci@1/pci@0/pci@e           /SYS/RCSA/PCIE7
    pci@480/pci@1/pci@0/pci@4           /SYS/RCSA/PCIE14
    pci@4c0/pci@1/pci@0/pci@8           /SYS/RCSA/PCIE15
```

```
    pci@4c0/pci@1/pci@0/pci@4        /SYS/RCSA/PCIE16
    pci@4c0/pci@1/pci@0/pci@c/pci@0/pci@c /SYS/MB/SASHBA1
    pci@4c0/pci@1/pci@0/pci@c/pci@0/pci@4 /SYS/RIO/NET2
VCC
    NAME             PORT-RANGE
    primary-vcc0     5000-5100
        CLIENT                     PORT   LOGGING
        was2@primary-vcc0          5001   on
        db2@primary-vcc0           5000   on
        db1@primary-vcc0           5002   on
        was1@primary-vcc0          5003   on

VSW
    NAME             MAC               NET-DEV   ID   DEVICE     LINKPROP   DEFAULT-
VLAN-ID PVID VID                  MTU   MODE   INTER-VNET-LINK
    primary-vsw0     00:14:4f:fb:56:f0 net0      0    switch@0              1
1                         1500         on
        PEER                       MAC               PVID VID                  MTU
LINKPROP   INTERVNETLINK
        vnet0@was2                 00:14:4f:fa:6c:10 1                       1500
        vnet0@db2                  00:14:4f:f9:b1:e0 1                       1500
        vnet0@db1                  00:14:4f:fa:98:5b 1                       1500
        vnet0@was1                 00:14:4f:f8:13:f2 1                       1500

VDS
    NAME             VOLUME         OPTIONS          MPGROUP        DEVICE
    primary-vds0     was2
/dev/zvol/dsk/ldompool/was2/zdisk0
                     s11_image
/root/Downloads/sol-11_1-text-sparc.iso
                     db2
/dev/zvol/dsk/ldompool/db2/zdisk0
                     s10_image
/root/Downloads/sol-10-u11-ga-sparc-dvd.iso
                     db1
/dev/zvol/dsk/ldompool/db1/zdisk0
                     was1
/dev/zvol/dsk/ldompool/was1/zdisk0
        CLIENT                     VOLUME
        vdisk_was2@was2            was2
        s11_image@was2             s11_image
        vdisk_db2@db2              db2
        s10_image@db2              s10_image
        vdisk_db1@db1              db1
        vdisk_was1@was1            was1

VCONS
    NAME             SERVICE                    PORT   LOGGING
                     UART

------------------------------------------------------------------------------
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
db1              active     -n----  5002    128   128G     0.1%  0.1%  12d 12h 36m

UUID
    7ec90122-b43e-eae2-e859-943f1ddb585f

MAC
    00:14:4f:f8:bd:99

HOSTID
    0x84f8bd99

CONTROL
    failure-policy=ignore
    extended-mapin-space=on
    cpu-arch=native
    rc-add-policy=
    shutdown-group=15

DEPENDENCY
    master=

CORE
        CID    CPUSET
    16     (128, 129, 130, 131, 132, 133, 134, 135)
    17     (136, 137, 138, 139, 140, 141, 142, 143)
    ...
    30     (240, 241, 242, 243, 244, 245, 246, 247)
```

```
    31      (248, 249, 250, 251, 252, 253, 254, 255)

VCPU
    VID     PID     CID     UTIL NORM STRAND
    0       16      0       0.3% 0.3%   100%
    1       17      0       0.1% 0.1%   100%
    ...
    127     126     15      0.1% 0.1%   100%
    128     127     15      0.1% 0.1%   100%

MEMORY
    RA              PA              SIZE
    0x80000000      0x1000000000    128G

CONSTRAINT
    cpu=whole-core
    max-cores=unlimited
    threading=max-throughput
    physical-bindings=core

VARIABLES
    auto-boot?=false
    boot-device=disk:a disk net
    boot-devices=vdisk_db1
    keyboard-layout=US-English
    pm_boot_policy=disabled=0;ttfc=2000;ttmr=0;

IO
    DEVICE                          PSEUDONYM       OPTIONS
    pci@480/pci@1/pci@0/pci@a       /SYS/RCSA/PCIE13
NETWORK
    NAME            SERVICE                     ID   DEVICE     MAC               MODE
PVID VID                MTU   LINKPROP
    vnet0           primary-vsw0@primary        0    network@0  00:14:4f:fa:98:5b
1                       1500
        PEER                        MAC               MODE   PVID VID
MTU   LINKPROP
        primary-vsw0@primary        00:14:4f:fb:56:f0        1
1500
        vnet0@was2                  00:14:4f:fa:6c:10        1
1500
        vnet0@db2                   00:14:4f:f9:b1:e0        1
1500
        vnet0@was1                  00:14:4f:f8:13:f2        1
1500

DISK
    NAME            VOLUME                      TOUT ID   DEVICE  SERVER
MPGROUP
    vdisk_db1       db1@primary-vds0                 0    disk@0  primary

VCONS
    NAME            SERVICE                     PORT   LOGGING
    db1             primary-vcc0@primary        5002   on

--------------------------------------------------------------------------------
NAME            STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
was1            active     -n----  5003    128   256G     0.2%  0.2%  12d 12h 36m

UUID
    e80b786e-a515-c508-bab4-e9c88231cd30

MAC
    00:14:4f:fa:ba:7b

HOSTID
    0x84faba7b

CONTROL
    failure-policy=ignore
    extended-mapin-space=on
    cpu-arch=native
    rc-add-policy=
    shutdown-group=15

DEPENDENCY
    master=

CORE
```

```
    CID    CPUSET
    32     (256, 257, 258, 259, 260, 261, 262, 263)
    33     (264, 265, 266, 267, 268, 269, 270, 271)
    ...
    46     (368, 369, 370, 371, 372, 373, 374, 375)
    47     (376, 377, 378, 379, 380, 381, 382, 383)

VCPU
    VID    PID    CID    UTIL NORM STRAND
    0      256    32     8.5% 8.5%   100%
    1      257    32     0.1% 0.1%   100%
   ...
    126    382    47     0.0% 0.0%   100%
    127    383    47     0.0% 0.0%   100%

MEMORY
    RA               PA               SIZE
    0x80000000       0x100000000000   256G

CONSTRAINT
    cpu=whole-core
    max-cores=unlimited
    threading=max-throughput
    physical-bindings=core,memory

VARIABLES
    auto-boot?=false
    boot-device=/virtual-devices@100/channel-devices@200/disk@0:a disk net
    boot-devices=vdisk_was1
    pm_boot_policy=disabled=0;ttfc=2000;ttmr=0;

IO
    DEVICE                          PSEUDONYM        OPTIONS
    pci@340/pci@1/pci@0/pci@6       /SYS/RCSA/PCIE3
NETWORK
    NAME             SERVICE                       ID   DEVICE     MAC               MODE
PVID VID                  MTU   LINKPROP
    vnet0            primary-vsw0@primary          0    network@0  00:14:4f:f8:13:f2
1                         1500
        PEER                        MAC               MODE   PVID VID
MTU   LINKPROP
        primary-vsw0@primary        00:14:4f:fb:56:f0        1
1500
        vnet0@was2                  00:14:4f:fa:6c:10        1
1500
        vnet0@db2                   00:14:4f:f9:b1:e0        1
1500
        vnet0@db1                   00:14:4f:fa:98:5b        1
1500

DISK
    NAME             VOLUME                        TOUT ID   DEVICE  SERVER
MPGROUP
    vdisk_was1       was1@primary-vds0                  0    disk@0  primary

VCONS
    NAME             SERVICE                       PORT   LOGGING
    was1             primary-vcc0@primary          5003   on
```

SPARC T4-4 Server Configuration

```
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
was              active     -n----  5000    64    128G      96%   96%  61d 21h 39m

UUID
    5b777131-5636-408d-f0ca-91b8aed8305d

MAC
    00:14:4f:f9:d3:bc

HOSTID
    0x84f9d3bc
```

```
CONTROL
    failure-policy=ignore
    extended-mapin-space=off
    cpu-arch=native
    rc-add-policy=
    shutdown-group=0

DEPENDENCY
    master=

CORE
    CID    CPUSET
    16     (128, 129, 130, 131, 132, 133, 134, 135)
    17     (136, 137, 138, 139, 140, 141, 142, 143)
    18     (144, 145, 146, 147, 148, 149, 150, 151)
    19     (152, 153, 154, 155, 156, 157, 158, 159)
    20     (160, 161, 162, 163, 164, 165, 166, 167)
    21     (168, 169, 170, 171, 172, 173, 174, 175)
    22     (176, 177, 178, 179, 180, 181, 182, 183)
    23     (184, 185, 186, 187, 188, 189, 190, 191)

VCPU
    VID    PID    CID    UTIL NORM STRAND
    0      128    16     2.7% 2.7%   100%
    1      129    16      74%  74%   100%
    ...
    62     190    23      65%  65%   100%
    63     191    23      67%  67%   100%

MEMORY
    RA               PA               SIZE
    0x80000000       0x400000000      8G
    0x400000000      0xc00000000      8G
    0x800000000      0x1400000000     8G
    0xc00000000      0x1c00000000     8G
    0x1000000000     0x2400000000     8G
    0x1400000000     0x2c00000000     8G
    0x1800000000     0x3400000000     8G
    0x1c00000000     0x3c00000000     8G
    0x2000000000     0x4400000000     8G
    0x2400000000     0x4c00000000     8G
    0x2800000000     0x5400000000     8G
    0x2c00000000     0x5c00000000     8G
    0x3000000000     0x6400000000     8G
    0x3400000000     0x6c00000000     8G
    0x3800000000     0x7400000000     8G
    0x3c00000000     0x7c00000000     8G

CONSTRAINT
    threading=max-throughput

VARIABLES
    pm_boot_policy=disabled=1;ttfc=0;ttmr=0;

IO
    DEVICE                            PSEUDONYM        OPTIONS
    pci@400/pci@2/pci@0/pci@1         /SYS/PCI-EM0
DISK
    NAME             VOLUME                       TOUT ID   DEVICE  SERVER
MPGROUP
    was              vol1@primary-vds0                 0    disk@0  primary
    vdisk4           vol4@primary-vds0                 1    disk@1  primary

VCONS
    NAME             SERVICE                      PORT   LOGGING
     was              primary-vcc0@primary        5000   on
```

Appendix B: Details for DB2

DB2 Database Creation

```
DB="tradedb"
BACKUP="/export/home/db2inst1/backup"
{
db2set DB2_APM_PERFORMANCE=
db2stop force
ipclean -a
db2start

db2 force application all

db2 drop db ${DB}
db2 restore db ${DB} from ${BACKUP} to /ramdiskdata0 replace existing

db2 update dbm cfg using notifylevel 0
db2 update dbm cfg using diaglevel 1
db2 update dbm cfg using NUM_POOLAGENTS 500 automatic MAX_COORDAGENTS 500 automatic
MAX_CONNECTIONS 500 automatic
db2 update dbm cfg using DFT_MON_BUFPOOL on DFT_MON_LOCK on DFT_MON_SORT on
DFT_MON_STMT on DFT_MON_TIMESTAMP on DFT_MON_UOW off  DFT_MON_TABLE on
db2 -v update db cfg for ${DB} using MAXLOCKS 100 LOCKLIST 100000

db2 update dbm cfg using dft_mon_lock on
db2 update dbm cfg using dft_mon_stmt on
db2 update dbm cfg using dft_mon_table on
db2 update dbm cfg using dft_mon_uow on

db2 connect to ${DB}
db2 update db cfg for ${DB} using maxappls 500 automatic
db2 update db cfg for ${DB} using logfilsiz 8000
db2 update db cfg for ${DB} using logprimary 10
db2 update db cfg for ${DB} using newlogpath /ramdisklog0
db2 update db cfg for ${DB} using dft_queryopt 0

db2 update db cfg for ${DB} using softmax 3000
db2 update db cfg for ${DB} using chngpgs_thresh 99

db2 -v alter bufferpool IBMDEFAULTBP size -1
db2 -v connect reset
db2 -v update db cfg for ${DB} using BUFFPAGE 262144

db2set DB2_APM_PERFORMANCE=ON
db2set DB2_KEEPTABLELOCK=CONNECTION
db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON
db2set DB2_MINIMIZE_LISTPREFETCH=YES

db2 connect reset
db2 terminate

db2stop force
db2start

db2 connect to ${DB}
db2 reorgchk update statistics
db2 connect reset
db2 terminate
```

DB2 Tuning

```
DB2 Database Configuration
       Database Configuration for Database

 Database configuration release level                  = 0x0f00
 Database release level                                = 0x0f00

 Database territory                                    = US
 Database code page                                    = 1208
 Database code set                                     = UTF-8
 Database country/region code                          = 1
 Database collating sequence                           = IDENTITY
 Alternate collating sequence              (ALT_COLLATE) =
 Number compatibility                                  = OFF
```

```
 Varchar2 compatibility                                = OFF
 Date compatibility                                    = OFF
 Database page size                                    = 4096

 Statement concentrator                     (STMT_CONC) = OFF

 Discovery support for this database       (DISCOVER_DB) = ENABLE

 Restrict access                                       = NO
 Default query optimization class         (DFT_QUERYOPT) = 0
 Degree of parallelism                      (DFT_DEGREE) = ANY
 Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
 Default refresh age                   (DFT_REFRESH_AGE) = 0
 Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
 Number of frequent values retained     (NUM_FREQVALUES) = 10
 Number of quantiles retained            (NUM_QUANTILES) = 20

 Decimal floating point rounding mode  (DECFLT_ROUNDING) = ROUND_HALF_EVEN

 Backup pending                                        = NO

 All committed transactions have been written to disk    = NO
 Rollforward pending                                   = NO
 Restore pending                                       = NO

 Multi-page file allocation enabled                    = YES

 Log retain for recovery status                        = NO
 User exit for logging status                          = NO

 Self tuning memory                    (SELF_TUNING_MEM) = ON
 Size of database shared memory (4KB)  (DATABASE_MEMORY) = AUTOMATIC(1328376)
 Database memory threshold               (DB_MEM_THRESH) = 10
 Max storage for lock list (4KB)                (LOCKLIST) = 100000
 Percent. of lock lists per application         (MAXLOCKS) = 100
 Package cache size (4KB)                    (PCKCACHESZ) = AUTOMATIC(8192)
 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = AUTOMATIC(113508)
 Sort list heap (4KB)                           (SORTHEAP) = AUTOMATIC(7945)

 Database heap (4KB)                              (DBHEAP) = AUTOMATIC(6887)
 Catalog cache size (4KB)               (CATALOGCACHE_SZ) = 300
 Log buffer size (4KB)                          (LOGBUFSZ) = 2154
 Utilities heap size (4KB)                 (UTIL_HEAP_SZ) = 524288
 Buffer pool size (pages)                       (BUFFPAGE) = 262144
 SQL statement heap (4KB)                       (STMTHEAP) = AUTOMATIC(8192)
 Default application heap (4KB)               (APPLHEAPSZ) = AUTOMATIC(256)
 Application Memory Size (4KB)                (APPL_MEMORY) = AUTOMATIC(40000)
 Statistics heap size (4KB)                  (STAT_HEAP_SZ) = AUTOMATIC(4384)

 Interval for checking deadlock (ms)           (DLCHKTIME) = 10000
 Lock timeout (sec)                           (LOCKTIMEOUT) = -1

 Changed pages threshold                   (CHNGPGS_THRESH) = 99
 Number of asynchronous page cleaners      (NUM_IOCLEANERS) = AUTOMATIC(112)
 Number of I/O servers                      (NUM_IOSERVERS) = AUTOMATIC(3)
 Index sort flag                                (INDEXSORT) = YES
 Sequential detect flag                         (SEQDETECT) = YES
 Default prefetch size (pages)            (DFT_PREFETCH_SZ) = AUTOMATIC

 Track modified pages                           (TRACKMOD) = NO

 Default number of containers                          = 1
 Default tablespace extentsize (pages)   (DFT_EXTENT_SZ) = 32

 Max number of active applications              (MAXAPPLS) = AUTOMATIC(500)
 Average number of active applications         (AVG_APPLS) = AUTOMATIC(1)
 Max DB files open per application              (MAXFILOP) = 61440

 Log file size (4KB)                           (LOGFILSIZ) = 8000
 Number of primary log files                  (LOGPRIMARY) = 10
 Number of secondary log files                 (LOGSECOND) = 12
 Changed path to log files                    (NEWLOGPATH) =
 Path to log files                                     =
/ramdisklog1/NODE0000/LOGSTREAM0000/
 Overflow log path                       (OVERFLOWLOGPATH) =
 Mirror log path                           (MIRRORLOGPATH) =
 First active log file                                 =
 Block log on disk full                 (BLK_LOG_DSK_FUL) = NO
 Block non logged operations             (BLOCKNONLOGGED) = NO
```

```
Percent max primary log space by transaction  (MAX_LOG) = 0
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

Percent log file reclaimed before soft chckpt (SOFTMAX) = 520

HADR database role                                      = STANDARD
HADR local host name                 (HADR_LOCAL_HOST) =
HADR local service name               (HADR_LOCAL_SVC) =
HADR remote host name                (HADR_REMOTE_HOST) =
HADR remote service name              (HADR_REMOTE_SVC) =
HADR instance name of remote server  (HADR_REMOTE_INST) =
HADR timeout value                       (HADR_TIMEOUT) = 120
HADR target list                     (HADR_TARGET_LIST) =
HADR log write synchronization mode     (HADR_SYNCMODE) = NEARSYNC
HADR spool log data limit (4KB)      (HADR_SPOOL_LIMIT) = 0
HADR log replay delay (seconds)     (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 0

First log archive method                 (LOGARCHMETH1) = OFF
Archive compression for logarchmeth1    (LOGARCHCOMPR1) = OFF
Options for logarchmeth1                  (LOGARCHOPT1) =
Second log archive method                (LOGARCHMETH2) = OFF
Archive compression for logarchmeth2    (LOGARCHCOMPR2) = OFF
Options for logarchmeth2                  (LOGARCHOPT2) =
Failover log archive path                (FAILARCHPATH) =
Number of log archive retries on error   (NUMARCHRETRY) = 5
Log archive retry Delay (secs)         (ARCHRETRYDELAY) = 20
Vendor options                               (VENDOROPT) =

Auto restart enabled                      (AUTORESTART) = ON
Index re-creation time and redo index build  (INDEXREC) = SYSTEM (RESTART)
Log pages during index build            (LOGINDEXBUILD) = OFF
Default number of loadrec sessions    (DFT_LOADREC_SES) = 1
Number of database backups to retain   (NUM_DB_BACKUPS) = 12
Recovery history retention (days)     (REC_HIS_RETENTN) = 366
Auto deletion of recovery objects    (AUTO_DEL_REC_OBJ) = OFF

TSM management class                    (TSM_MGMTCLASS) =
TSM node name                            (TSM_NODENAME) =
TSM owner                                   (TSM_OWNER) =
TSM password                             (TSM_PASSWORD) =

Automatic maintenance                       (AUTO_MAINT) = ON
  Automatic database backup              (AUTO_DB_BACKUP) = OFF
  Automatic table maintenance            (AUTO_TBL_MAINT) = ON
    Automatic runstats                    (AUTO_RUNSTATS) = ON
      Real-time statistics              (AUTO_STMT_STATS) = ON
      Statistical views                (AUTO_STATS_VIEWS) = OFF
      Automatic sampling                  (AUTO_SAMPLING) = OFF
    Automatic statistics profiling     (AUTO_STATS_PROF) = OFF
      Statistics profile updates         (AUTO_PROF_UPD) = OFF
    Automatic reorganization                (AUTO_REORG) = OFF

Auto-Revalidation                           (AUTO_REVAL) = DEFERRED
Currently Committed                         (CUR_COMMIT) = ON
CHAR output with DECIMAL input         (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations         (ENABLE_XMLCHAR) = YES
WLM Collection Interval (minutes)      (WLM_COLLECT_INT) = 0
Monitor Collect Settings
Request metrics                        (MON_REQ_METRICS) = BASE
Activity metrics                       (MON_ACT_METRICS) = BASE
Object metrics                         (MON_OBJ_METRICS) = EXTENDED
Routine data                               (MON_RTN_DATA) = NONE
  Routine executable list             (MON_RTN_EXECLIST) = OFF
Unit of work events                        (MON_UOW_DATA) = NONE
  UOW events with package list         (MON_UOW_PKGLIST) = OFF
  UOW events with executable list     (MON_UOW_EXECLIST) = OFF
Lock timeout events                    (MON_LOCKTIMEOUT) = NONE
Deadlock events                            (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events                           (MON_LOCKWAIT) = NONE
Lock wait event threshold                (MON_LW_THRESH) = 5000000
Number of package list entries          (MON_PKGLIST_SZ) = 32
Lock event notification level          (MON_LCK_MSG_LVL) = 1
```

```
 SMTP Server                                 (SMTP_SERVER) =
 SQL conditional compilation flags           (SQL_CCFLAGS) =
 Section actuals setting                (SECTION_ACTUALS) = NONE
 Connect procedure                         (CONNECT_PROC) =
 Adjust temporal SYSTEM_TIME period (SYSTIME_PERIOD_ADJ) = NO
 Log DDL Statements                       (LOG_DDL_STMTS) = NO
 Log Application Information              (LOG_APPL_INFO) = NO
 Default data capture on new Schemas    (DFT_SCHEMAS_DCC) = NO
 Database is in write suspend state                       = NO
```

DB2 Database Manager Configuration

```
Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0f00

 CPU speed (millisec/instruction)             (CPUSPEED) = 2.400414e-07
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) = 1.000000e+02

 Max number of concurrently active databases     (NUMDB) = 32
 Federated Database System Support           (FEDERATED) = NO
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account           (DFT_ACCOUNT_STR) =

 Java Development Kit installation path       (JDK_PATH) =
/export/home/db2inst1/sqllib/java/jdk64

 Diagnostic error capture level              (DIAGLEVEL) = 1
 Notify Level                               (NOTIFYLEVEL) = 0
 Diagnostic data directory path                (DIAGPATH) =
/export/home/db2inst1/sqllib/db2dump/
 Current member resolved DIAGPATH                         =
/export/home/db2inst1/sqllib/db2dump/
 Alternate diagnostic data directory path (ALT_DIAGPATH) =
 Current member resolved ALT_DIAGPATH                     =
 Size of rotating db2diag & notify logs (MB)  (DIAGSIZE) = 0

 Default database monitor switches
   Buffer pool                           (DFT_MON_BUFPOOL) = ON
   Lock                                     (DFT_MON_LOCK) = ON
   Sort                                     (DFT_MON_SORT) = ON
   Statement                                (DFT_MON_STMT) = ON
   Table                                   (DFT_MON_TABLE) = ON
   Timestamp                          (DFT_MON_TIMESTAMP) = ON
   Unit of work                              (DFT_MON_UOW) = ON
 Monitor health of instance and databases   (HEALTH_MON) = OFF

 SYSADM group name                          (SYSADM_GROUP) = DB2IADM1
 SYSCTRL group name                        (SYSCTRL_GROUP) =
 SYSMAINT group name                      (SYSMAINT_GROUP) =
 SYSMON group name                          (SYSMON_GROUP) =

 Client Userid-Password Plugin           (CLNT_PW_PLUGIN) =
 Client Kerberos Plugin                 (CLNT_KRB_PLUGIN) =
 Group Plugin                              (GROUP_PLUGIN) =
 GSS Plugin for Local Authorization    (LOCAL_GSSPLUGIN) =
 Server Plugin Mode                    (SRV_PLUGIN_MODE) = UNFENCED
 Server List of GSS Plugins      (SRVCON_GSSPLUGIN_LIST) =
 Server Userid-Password Plugin         (SRVCON_PW_PLUGIN) =
 Server Connection Authentication           (SRVCON_AUTH) = NOT_SPECIFIED
 Cluster manager                                          =

 Database manager authentication         (AUTHENTICATION) = SERVER
 Alternate authentication           (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
 Cataloging allowed without authority    (CATALOG_NOAUTH) = NO
 Trust all clients                       (TRUST_ALLCLNTS) = YES
 Trusted client authentication           (TRUST_CLNTAUTH) = CLIENT
 Bypass federated authentication             (FED_NOAUTH) = NO

 Default database path                        (DFTDBPATH) = /export/home/db2inst1

 Database monitor heap size (4KB)           (MON_HEAP_SZ) = AUTOMATIC(90)
 Java Virtual Machine heap size (4KB)      (JAVA_HEAP_SZ) = 2048
 Audit buffer size (4KB)                   (AUDIT_BUF_SZ) = 0
 Size of instance shared memory (4KB)  (INSTANCE_MEMORY) = AUTOMATIC(30732852)
```

```
Instance memory for restart light (%) (RSTRT_LIGHT_MEM) = AUTOMATIC(10)
Agent stack size                       (AGENT_STACK_SZ) = 1024
Sort heap threshold (4KB)                  (SHEAPTHRES) = 0

Directory cache support                     (DIR_CACHE) = YES

Application support layer heap size (4KB)   (ASLHEAPSZ) = 15
Max requester I/O block size (bytes)         (RQRIOBLK) = 32767
Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

Priority of agents                           (AGENTPRI) = SYSTEM
Agent pool size                         (NUM_POOLAGENTS) = AUTOMATIC(500)
Initial number of agents in pool        (NUM_INITAGENTS) = 0
Max number of coordinating agents      (MAX_COORDAGENTS) = AUTOMATIC(500)
Max number of client connections       (MAX_CONNECTIONS) = AUTOMATIC(500)

Keep fenced process                         (KEEPFENCED) = YES
Number of pooled fenced processes          (FENCED_POOL) = AUTOMATIC(MAX_COORDAGENTS)
Initial number of fenced processes     (NUM_INITFENCED) = 0

Index re-creation time and redo index build  (INDEXREC) = RESTART

Transaction manager database name          (TM_DATABASE) = 1ST_CONN
Transaction resync interval (sec)      (RESYNC_INTERVAL) = 180

SPM name                                     (SPM_NAME) = p3519_06
SPM log size                           (SPM_LOG_FILE_SZ) = 256
SPM resync agent limit                  (SPM_MAX_RESYNC) = 20
SPM log path                              (SPM_LOG_PATH) =

TCP/IP Service name                          (SVCENAME) = db2c_db2inst1
Discovery mode                               (DISCOVER) = SEARCH
Discover server instance                (DISCOVER_INST) = ENABLE

SSL server keydb file                    (SSL_SVR_KEYDB) =
SSL server stash file                    (SSL_SVR_STASH) =
SSL server certificate label             (SSL_SVR_LABEL) =
SSL service name                           (SSL_SVCENAME) =
SSL cipher specs                        (SSL_CIPHERSPECS) =
SSL versions                               (SSL_VERSIONS) =
SSL client keydb file                    (SSL_CLNT_KEYDB) =
SSL client stash file                    (SSL_CLNT_STASH) =

Maximum query degree of parallelism    (MAX_QUERYDEGREE) = ANY
Enable intra-partition parallelism      (INTRA_PARALLEL) = NO

Maximum Asynchronous TQs per query     (FEDERATED_ASYNC) = 0

No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = AUTOMATIC(4096)
No. of int. communication channels   (FCM_NUM_CHANNELS) = AUTOMATIC(2048)
Inter-node comm. parallelism           (FCM_PARALLELISM) = 1
Node connection elapse time (sec)          (CONN_ELAPSE) = 10
Max number of node connection retries (MAX_CONNRETRIES) = 5
Max time difference between nodes (min) (MAX_TIME_DIFF) = 60

db2start/db2stop timeout (min)         (START_STOP_TIME) = 10

WLM dispatcher enabled                  (WLM_DISPATCHER) = NO
WLM dispatcher concurrency             (WLM_DISP_CONCUR) = COMPUTED
WLM dispatcher CPU shares enabled (WLM_DISP_CPU_SHARES) = NO
WLM dispatcher min. utilization (%) (WLM_DISP_MIN_UTIL) = 5

Communication buffer exit library list (COMM_EXIT_LIST) =
```

An Analysis of Performance, Scaling, and Best Practices for IBM WebSphere Application Server on Oracle's SPARC T-Series Servers
September 2014, Version 1.0
Authors: E. Reid, N. Qi, and K. Arhelger

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com