

# Availability Modeling for the Sun™ Java System Application Server Enterprise Edition 7

Technical White Paper





# Table of Contents

Table of Contents.....	1
Executive Summary.....	2
1. Methodology and Tools.....	4
Availability Overview.....	4
RAScad.....	4
2. System Configuration.....	6
High Availability Database – HADB.....	7
3. Assumptions.....	9
Faults Modeled.....	10
Critical Items Not Modeled.....	11
4. Model Parameters.....	12
HADB Node Parameters.....	12
Application Server Instance Parameters.....	12
HADB Restart and Repair Time.....	13
Fraction of Imperfect Recovery .....	13
HADB Restore Time .....	13
Session Recovery Time.....	13
Application Server Restart Time.....	14
Application Server Restore Time.....	14
Summary – MTTR Parameters.....	14
5. Model Structure.....	15
6. Analysis of Results.....	18
7. Conclusions.....	22
References.....	23
More Information.....	24

## Executive Summary

Reliability, availability, and serviceability (RAS) are becoming increasingly important for networked computer server and storage systems running business-critical applications. RAS is one of the major issues considered in designing and deploying enterprise products and services. This paper presents an availability analysis for multitier, Java™ and Web-based application deployments running on the Sun™ Java System Application Server Enterprise Edition 7 software. The results show that this software is able to support 99.999-percent (referred to as five-nines) application availability in common Web deployment scenarios, assuming that the software is also supported by reliable database and load balancer tiers.

The analysis also reveals that an optimal configuration for availability—achieving well under five minutes downtime per year—is four instances of the Sun Java System Application Server Enterprise Edition 7 software and four pairs of Highly Available Database (HADB) nodes (HADB is a reliable database architecture employed for session replication). Furthermore, a sensitivity analysis indicates that customers can obtain higher levels of availability by using recommended configurations of the Sun Java System Application Server Enterprise Edition 7 software coupled with careful attention to application design, quality, and deployment architecture. This conservative model relied on parameters constructed with field data. However, the performance of field systems may vary depending on the application quality, load characteristics, deployment architecture, hardware and network reliability, electrical power, environmental hazards, maintenance events not executed according to documented procedures, and certain other factors described in this paper.

The Sun Java System Application Server Enterprise Edition 7 software delivers exceptional availability beyond five-nines, as demonstrated in this paper, and summarized in Table 1.

Table 1: Overview of Availability for Configurations of the Sun Java System Application Server Enterprise Edition 7 Software

Configuration Description: Application Server Instances and HADB Pairs	Availability	Yearly Downtime
1 instance, 0 pairs	99.9629%	3 hours, 15 minutes
2 instances, 2 pairs	99.9993%	3 minutes, 30 seconds
4 instances, 4 pairs	99.9995%	2 minutes, 18 seconds
6 instances, 6 pairs	99.9993%	3 minutes, 26 seconds
8 instances, 8 pairs	99.9991%	4 minutes, 35 seconds
10 instances, 10 pairs	99.9989%	5 minutes, 44 seconds

System architects may find the information in this paper useful when designing enterprise applications based on Web and Java technology. The information is presented as follows:

- Chapter One: Methodology and tools
- Chapter Two: System configurations
- Chapter Three: Assumptions
- Chapter Four: Model parameters as well as various factors and values used in this analysis
- Chapter Five: Model structure and how each element affects the overall availability
- Chapter Six: Results, including sensitivity analysis
- Chapter Seven: Conclusions
- References

# 1. Methodology and Tools

## Availability Overview

A number of factors affect availability. Though largely dependent on the application and user population, a typical definition of availability is:

The percentage of time the system can perform its primary function within predefined quality of service (QoS) limits.

A standard formula for determining the planned availability of a component or system is:

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

This formula calculates availability where A is the availability of the system, MTTF is the mean time to failure, and MTTR is the mean time to recovery. According to the formula, availability is calculated as the proportion of uptime for a given period, taking into account the time it typically requires for the system to recover from unplanned failures and planned upgrades. Two implications of this formula are:

- A large MTTF makes the MTTR less relevant. For example, if a component has a MTTF of 10 years, a large MTTR—say five hours—may be acceptable, since downtime is infrequent.
- A small MTTR can often compensate for a small MTTF. For example, if a component fails frequently—say once a day—but recovers within milliseconds, the failures may not be noticeable (assuming the failures do not result in the loss of non-recoverable state).

## RAScad

Previous studies have shown that in many cases, it is possible to use a combination of measurement and mathematical models to derive a quantitative assessment of a software system. The Hsueh<sup>1</sup> study was the first to use Markov reward models combined with operational failure data to model an operating system (IBM/MVS). Later, similar techniques were applied to the Tandem Guardian and VAX/VMS operating systems<sup>2</sup>. The methodology was further extended from using operational data to using test data in evaluating availability for air traffic control software systems<sup>3</sup>. These studies all rely on failure data in estimating parameters that are plugged into the models. The models then can be solved to generate system availability or performance measures.

Methods to estimate parameters and associated confidence levels, including situations in which failure was not observed during the measurement period, have been addressed<sup>3,4</sup>.

The Markov reward model is one of the most powerful and widely accepted mathematical models in availability and reliability analysis<sup>5,6</sup>. The state space-based model structure and reward rate associated with each state in Markov reward models provide capabilities to evaluate availability, performance, service cost, and various metrics of interest for the modeled system. However, in modeling a real system, the number of states in the model often exceeds the range that can be handled manually and introduces the state explosion problem. To reduce model complexity, the hierarchical modeling approach has been proposed. It decomposes a complex model into multiple submodels and implements them in commercial modeling tools<sup>7,8</sup>.

The Sun software tool used in this availability analysis is RAScad<sup>9</sup>, a Web-based RAS architecture modeling tool for system design and development phases. It helps model the availability in complex systems by specifying parameters that affect MTTF and MTTR. RAScad integrates two modules: Model Generator (MG), which provides automatic model generation specific to Sun product RAS characteristics; and Graphical Model Builder (GMB), which provides general graphical Markov, semi-Markov, and reliability block diagram modeling capabilities. Both MG and GMB support the hierarchical modeling approach.

RAScad has been heavily used in designing new Sun hardware products. It has also been used to develop availability models for Sun Cluster software systems<sup>10</sup>. RAS metrics generated by the tool include availability, performance, system interruption rate, and system service call rate. The model presented in this paper was developed using the RAScad GMB module, and the system metrics of interest are availability, associated yearly downtime, and MTTF.

## 2. System Configuration

Figure 1 shows a general configuration for the Sun Java System Application Server Enterprise Edition 7 product, which is analyzed in this paper. Business logic executes on multiple application server instances. Conversational state of the Java 2 Platform, Enterprise Edition (J2EE™) technology-based applications is written to an HADB<sup>11</sup>. Multiple pairs of HADB nodes are organized as Database Redundancy Units (DRU) for storing session persistence data. The test configuration also uses a relational database and Lightweight Directory Access Control (LDAP) server.

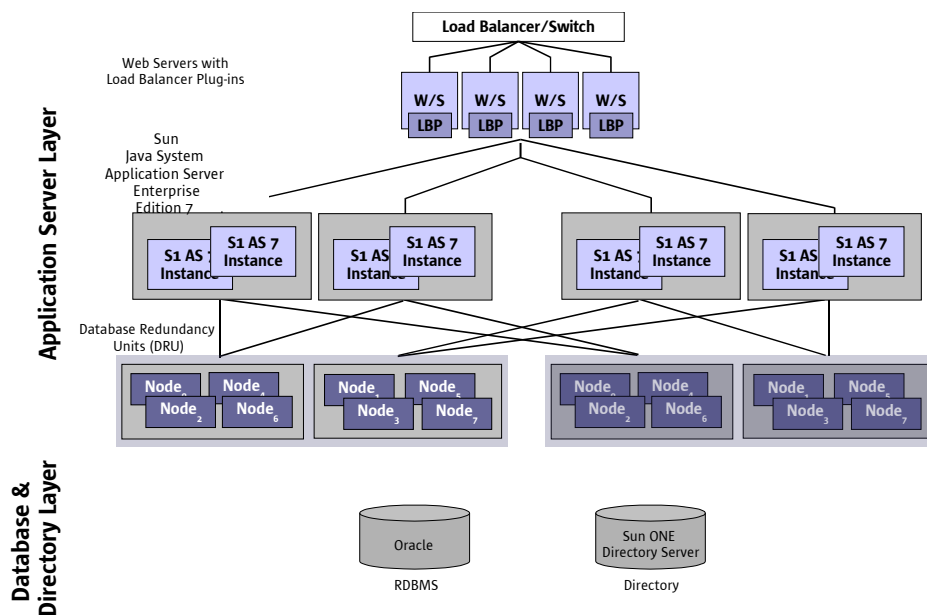


Figure 1. General Configuration of the Sun Java System Application Server Enterprise Edition 7 Software as Used in These Tests.

Longevity test results were obtained using two large applications. The first application is a sophisticated, real-world J2EE technology-based Web application for running digital marketplaces. It includes Catalog, Auction, RFP, Pricing, and Order Management modules, and is used in many customer deployments. The application is built with JavaServer Pages™ (JSP™) and Java Servlet technology, and uses pooled JDBC™ technology to access an Oracle database. It also accesses the Sun Java System Directory Server product, utilizing the Java LDAP Software Developer Kit (SDK). The average session size is 50 KB, which is larger than the typical HTTP session size.

The other test application is the Nile Bookstore application benchmark built with JavaServer Pages™ (JSP™) and Java Servlet technology, which uses the Sun Java System Application Server Enterprise Edition 7 connection pooling capabilities to access an

Oracle database. The average session size in this application is roughly 30 KB. This application is a complete, end-to-end, e-commerce application server benchmark that has been widely used by independent testing laboratories, including Doculabs, eWeek, and PC Magazine, to evaluate application server products. Both applications were deployed on the Sun Java System Application Server Enterprise Edition 7 software.

## High Availability Database – HADB

The Sun Java System Application Server Enterprise Edition 7 product supports persistence of HTTP session data, HTTPS credentials, Stateful Session Bean state and Entity Bean primary key in its bundled High Availability Database (HADB). The high-performance HADB is a robust, scalable, available, and distributed database that is capable of operating in excess of five-nines availability, making this a Class 5 component. The HADB consists of two DRUs, which are logical groupings of nodes. The nodes in each DRU contain the complete set of session data. Nodes in a DRU are distributed over one or more physical hosts, but DRUs should never contain nodes that share a physical host.

A node is a collection of processes, a dedicated area of main memory, and some physical disk space. The HADB has several different types of nodes, the most common being the data node, which is used for storing and updating data. Data nodes may be active—that is, they contain data; or they may be spare nodes—that is, ready to take over for the failure of an active data node. Session data is fragmented across the nodes in each DRU so that all nodes are evenly balanced. This ensures optimal throughput and response time.

Session data is persisted by the Application Server to the HADB. Durability and reliability is provided by the patented Always-On availability technology<sup>11</sup> underlying HADB. Each session is stored in two nodes: a primary replica and a hot standby replica. The data is replicated asynchronously between nodes, but the replication is transactional. Data is always replicated across DRUs to ensure that each DRU holds a complete copy of the database.

In addition to providing a high degree of data availability through replication, the nodes that maintain and provide access to data are also highly fault-tolerant. Nodes work in pairs (mirrors) to provide a mutual watchdog service. On node failure, the mirror takes ownership of the failed node's primary fragments and continues to provide data access and updates.

Simultaneously, the failed node attempts recovery by trying to recover data from the active node using two different techniques. If successful, the recovered node takes back ownership of the primary fragments, returning the system to its prefailure configuration. To maintain high availability and capacity, recovery is extremely quick (often within 500 ms). If a failed node cannot be repaired, a spare node attempts to take its place, and the failed node becomes a spare.

Both of the test applications were stable. No memory or connection leaks were observed. During the course of the longevity test, application redeployment and server restart were not necessary. A load balancer was also included in the test configuration to perform sticky, round-robin load balancing between multiple application server instances.

In our test, the Sun Java System Application Server Enterprise Edition 7 instances ran on two and four-CPU Sun Enterprise™ 450 servers, and the HADB nodes were hosted on four Sun Ultra™ 80 workstations. Figure 2 shows the test configuration and logical relationship between layers. System load was generated through the SilkPerformer<sup>12</sup> load generator tool hosted on a Microsoft Windows machine. In these tests, the systems were given a capacity load of 60 to 70 percent, and multiple seven-day duration runs were performed. Load was kept constant throughout the test. One of the tests—which was continued for 24 days for sanity checking and availability demonstration purposes—survived a system reboot due to a recoverable hardware problem.

Table 2: Test Environment

<b>Load Balancers</b>	
<b>Sun Java System Application Server Enterprise Edition 7 #1</b>	<b>Sun Java System Application Server Enterprise Edition 7, #2</b>
<b>J2EE Web Application Nile Bookstore</b>	<b>J2EE Web Application Nile Bookstore</b>
<b>HADB 1 (2 Nodes)</b>	<b>HADB 2 (2 Nodes)</b>
<b>Oracle, Sun Java System Directory Server</b>	
<b>Solaris™ 9 OS running on Sun Enterprise 450 server</b>	

A number of fault injection tests were performed to ensure that the system could tolerate single faults, as expected in the architecture design. The following tests were performed:

- Selected HADB node brought down by manually killing HADB processes
- Communication within an HADB node pair disrupted by unplugging network cable from selected node
- HADB node hardware power unplugged
- HADB processes manually killed
- Sun Java System Application Server node brought down by manually killing its node
- Sun Java System Application Server node host network cable unplugged
- Sun Java System Application Server node host power unplugged

For all listed fault injection tests, the system continued functioning without major departures from expected performance. Roughly seven million requests were processed by the system in each run. These faults provided data on system recovery times, HADB restart times, restore times, and other model parameters. This data was used, in addition to field and design data, in constructing the analysis model in RAScad. See Model Parameters in chapter 4 for more information.

### 3. Assumptions

Although the test environment is a fixed configuration consisting of two Sun Java System Application Server, Enterprise Edition instances and two HADB node pairs, the modeled configurations are not limited to the test environment. Specifically, this analysis uses two common configurations:

- Configuration 1: Two Sun Java System Application Server instances, two HADB node pairs, and two HADB spare nodes (Figure 2).
- Configuration 2: Four Sun Java System Application Server instances, four HADB node pairs, and four HADB spare nodes (Figure 3).

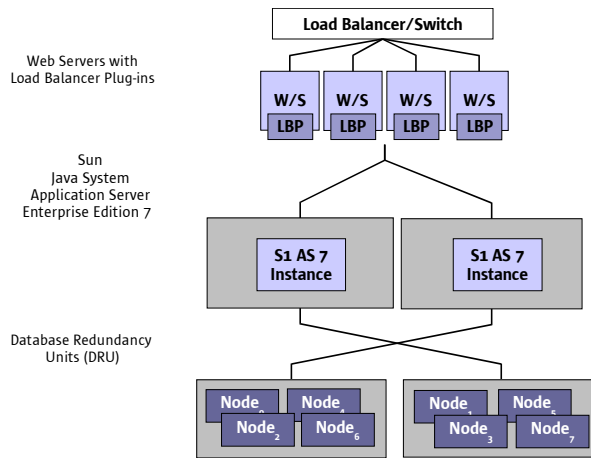


Figure 2. Diagram of Configuration 1

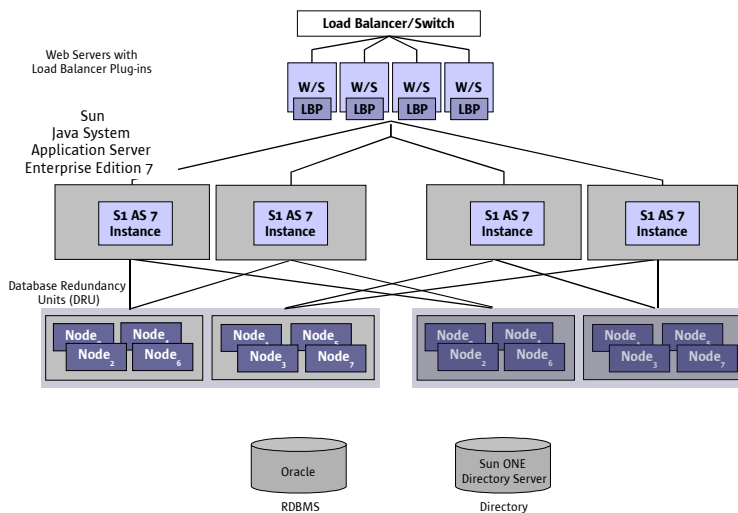


Figure 3. Diagram of Configuration 2

The modeled system is considered available if at least one application server instance is up and able to service requests, and the system is able to persist session state. Translating this definition to the model specification requires at least one application server instance and one node in each HADB node pair to be in the working state. The requirement for the HADB node pairs is based on each application server potentially using all the HADB node pairs, since the data table is fragmented across all node pairs due to data partitioning.

## Faults Modeled

Both hardware permanent faults and hardware or software transient faults occurring on all computer systems in the configuration were modeled. When a permanent hardware failure occurs, that computer system must be shut down for repair. A hardware failure on an application server node puts the affected server instances out of use until repairs are completed. For hardware failures on an HADB node used for state replication, the surviving companion node initiates repair by invoking a spare node.

Transient faults due to hardware or software problems have two possible results:

- Restart the applications (session persistence processes or the Sun Java System Application Server process) without a system reboot. This event is an HADB failure if it occurs on an HADB node, or it is an AS failure if it occurs on an application server instance.
- Or, reboot the operating system and perform a cold restart of all processes. This event is an OS failure.

Failure and repair processes are assumed to be independent on different computer systems. This assumption implies, for the Sun Java System Application Server Enterprise Edition 7 product, that the failure and restart of an application server instance does not introduce a failure on another application server instance. Because of this independence, the interaction between the surviving application server instance and the recovering application server instance is minimal, and imperfect recovery has no effect on the surviving node except to possibly increase recovery time. However, when an HADB node pair is in recovery mode, the surviving companion node is actively participating in the recovery process by transferring data updated during the outage to the recovering node, and thus a possible but highly uncommon mode of failure due to imperfect recovery is also modeled.

It is assumed that the failure rate is constant (exponential distribution) for the purpose of steady-state analysis. In the sensitivity analysis discussed later in this document, all the failure rates in given ranges are changed to quantify the impact of difference failure rates on availability.

In addition, the failure rate/workload dependency is modeled as follows: After an HADB node or application server instance fails, the failure rate on the complementary HADB node or remaining application server instances is doubled to reflect the possibility of increased failure rate due to increased load. This is modeled as follows:

If  $La_0$  denotes the base failure rate for an application server instance, the application server failure rate after the  $i$ th instance has failed is:  $La_i = La_0 \times 2^i$ .

This model is based on the observation that the risk of software failure increases exponentially with increasing workload<sup>13</sup>.

### Potentially Critical Items Not Modeled

Several aspects of a physical deployment are beyond the scope of the current model. To simplify the study, the following elements are assumed not to fail:

- Database and directory servers (Oracle, LDAP, and so on)
- Web server and load balancer
- Electrical power
- Network communication
- HVAC (Heating, ventilation, and air conditioning)

We assume that in a typical data center well-engineered and commercial-quality solutions for the above elements are already in place.

No real failures of the above were observed during the study, and no artificial failures were injected in the above areas (though errors such as power loss and network connection loss were injected into the application server layer). This simplification allowed us to focus on characterizing the availability of the Sun Java System Application Server Enterprise Edition 7 software stack and Session Replication (DRU) layers.

It should be emphasized that human error, which is NOT considered in the model, could be critical to system availability. Historical data from many sources suggest that human error accounted for roughly 50 percent of all outages in production server environments during the past three decades<sup>14</sup>. Although redundancy provisions in the Sun Java System Application Server Enterprise Edition 7 software could tolerate a human error-induced failure on an application server instance or HADB node, the product is not designed to prevent catastrophic failure from human error introduced during online maintenance when redundancy may become temporarily unavailable. Therefore, it is assumed that product documentation is strictly followed and no operator errors occur. An administrator can orchestrate online upgrades to applications, the Sun Java System Application Server Enterprise Edition 7 software, the underlying operating system, and the hardware using single- or dual-cluster deployments. This model is restricted to simple, one-cluster deployments. It models four scheduled maintenance events for the HADB nodes. It is possible to extend this hierarchical model to include more events and subcomponents, but the model would become too complex to present in a white paper.

The intent is to prove that when all system components in the deployment are well configured, the Sun Java System Application Server Enterprise Edition 7 software supports very high levels of availability. We used the longevity runs to obtain some of the parameters used in the availability model. In the model, we also account for many interesting scenarios such as hardware failures which did not occur in our tests, but have been characterized in other reliability studies.

## 4. Model Parameters

The following parameters are used in this model. Lambda (and  $\lambda$ ) is used to represent mean failure rate, and  $\mu$  represents repair rate.

### HADB Node Parameters

- Node failure rate = 4/year
  - HADB (restartable) failure rate:  $\lambda_{\text{hadb}} = 2/\text{year}$
  - OS failure rate:  $\lambda_{\text{os}} = 1/\text{year}$
  - HW failure rate:  $\lambda_{\text{hw}} = 1/\text{year}$
- Restart Time
  - Short restart time (for HADB failure):  $T_{\text{start\_short}} = 1$  minute
  - Long restart time (for OS failure):  $T_{\text{start\_long}} = 15$  minutes
  - Repair time (for hardware failure):  $T_{\text{repair}} = 30$  minutes
- Fraction of Imperfect Recovery:  $\text{FIR} = 0.1\%$
- Maintenance Event
  - Maintenance rate:  $\lambda_{\text{mnt}} = 4/\text{year}$
  - Maintenance switchover time:  $T_{\text{mnt}} = 1$  minute
- Restore Time:  $T_{\text{restore}} = 1$  hour

### Application Server Instance Parameters

- Application Server Instance Failure Rate:  $\lambda = 52/\text{year}$ 
  - Application server (restartable) failure rate:  $\lambda_{\text{as}} = 50/\text{year}$
  - OS failure rate:  $\lambda_{\text{os}} = 1/\text{year}$
  - Hardware failure rate:  $\lambda_{\text{hw}} = 1/\text{year}$
- Failover Time:  $T_{\text{recovery}} = 5$  seconds
- Restart Time
  - Short restart time (for application server failure):  $T_{\text{start\_short}} = 90$  seconds
  - Long restart time (average for OS/hardware failure):  $T_{\text{start\_long}} = 1$  hours
    - Outage time for hardware failure: 100 minutes
    - Outage time for OS failure: 15 minutes
- Restore Time:  $T_{\text{start\_all}} = 30$  minutes

Conservative parameters are based on test or field data. For example, many recovery times are longer than those measured from test or field results. To tolerate variance on these parameters due to different hardware and software configurations, user workloads, and environmental factors (such as temperature), we conducted extensive parametric and uncertainty analysis (in the analysis section) on selected parameters.

### HADB Restart and Repair Time

When an HADB node suffers a failure, the database automatically tries to restart the failed node. The restart may be successful (for HADB and OS failures), or unsuccessful (for hardware failures). In the first case, the time from the detection of the failure to completion of the restart is the restart time. Although the measured restart time is only around 40 seconds, the model uses a more conservative value of one minute. The second case of an unsuccessful restart initiates an automatic repair process. The repair process copies all data and log records from the complementary node in the same pair to a spare node, and then rebuilds the data fragments on the spare node. The time taken to complete this procedure is the repair time, which depends on the amount of data stored on the node. It is estimated that the size of data on an HADB node is within one gigabyte, which can support up to 10,000 concurrent sessions with the average size of 50 KB on each application server instance (in a configuration of four application server instances). Measurements on a particular configuration show it takes 12 minutes to copy one gigabyte of data from one node to another. The model sets this parameter to 30 minutes, to accommodate possible variance on different configurations.

### Fraction of Imperfect Recovery

The HADB automatic recovery (restart or repair) process may not be perfect due to various unexpected situations, such as defects in the fault-handling software running on the complementary node, or latent faults on the disk activated during data reconstruction. Thus, there is a small chance that the complementary node could fail during the recovery process, resulting in system failure. Although the probability of such an event is very low, a Fraction of Imperfect Recovery (FIR) parameter was used to model the event. Based on the fact that imperfect recovery was not observed for more than 3,287 fault injections covering various failure scenarios, FIR is estimated to be below 0.1 percent at the 95-percent confidence level, using statistical functions for estimating upper and lower limits for the mean of a binomial random variable<sup>4</sup>.

### HADB Restore Time

When both nodes in an HADB node pair are down (a rare event), all session data supported by the pair is irretrievably lost, resulting in catastrophic failure. Human intervention is required to recreate a functional HADB node pair and system. The time to complete this procedure, restore time, includes time to notice failure, plus HADB recreation time. For 24x7 onsite maintenance, restore time is estimated to be one hour. During this time, the system is not available.

### Session Recovery Time

When an application server instance experiences a failure, user requests originally serviced by this instance are forwarded to other working instances. Average response time for the first request after failover is increased due to time spent reestablishing the session on another instance. This time increment is defined as session recovery time. Although session recovery time is measured at the subsecond levels, it is conservatively set to five seconds in the model.

### Application Server Restart Time

When an application server instance suffers a failure, it is automatically restarted on the same computer, as long as a system reboot is not required as part of the recovery process. The time to complete this procedure is the short restart time. An OS failure case requires a system reboot, conservatively assumed to take no more than 15 minutes. A hardware failure case requires a physical repair action (assumed to be 100 minutes, based on field data). Thus, the average restart time for the hardware/OS failure is approximately one hour, which is called long restart time.

### Application Server Restore Time

When all application server instances are down (a rare event), human intervention is required to restart the application servers. The restore time includes time to notice failure plus actual time to restart all application server instances. For 24x7 onsite maintenance, average application server restart time is assumed to be 30 minutes. This could be reduced to a very small number by using the Sun Cluster Data Service for the Sun Java System Application Server Enterprise Edition 7 software, but that solution is not considered in this model.

### Application Server Failure Rate

The failure rate on an AS node is conservatively set to once a week (including HW and OS failures), which is higher than upper bounds estimated based on the longest duration stress test we conducted. Given that no failure was observed during a 24-day test for two AS instances, a failure rate upper bound is estimated to be 1/16 day at the 95-percent confidence level and to be 1/9 day at the 99.5-percent confidence level, using statistical functions to estimate confidence intervals for the rate of failure with an exponential arrival time<sup>4</sup>.

### Summary—MTTR Parameters

Parameters such as HADB Restart and Repair Time, HADB Restore Time, Session Recovery Time, Application Server Restart Time, Fraction of Imperfect Recovery, and Application Server Restore Time all effect MTTR or MTTF. Overall availability can be increased by reducing MTTR or increasing MTTF, according to the availability model cited in the Methodology and Tools section:  $A = \text{MTTF}/(\text{MTTF}+\text{MTTR})$ . Parameters that increase MTTR, such as longer restore times, will decrease availability. Conversely, factors that decrease failure rate ( $1/\text{MTTF}$ ), such as a reduced chance of FIR, can increase availability.

## 5. Model Structure

This availability analysis models the Sun Java System Application Server Enterprise Edition 7 product in RAScad, including the HADB (DRU) subcomponent. RAScad is used to conservatively predict overall availability based on Markov’s hierarchical state transition model. The model employs MTTR and MTTF parameters. MTTF parameters include HADB failure rate, application server failure rate, OS failure rate, hardware failure rate, as well as restart, recovery, and repair times.

The hierarchical model consists of three state transition diagrams. Figure 3 models the overall Sun Java System Application Server Enterprise Edition 7 system as a three-state Markov model. Each state has a number, which represents the reward rate, associated with it. A reward rate of one (1) represents a working state. A reward rate of zero (0) represents a failure state. The notation of these states is listed below:

- Ok: At least one node in each HADB node pair is functioning properly, and at least one application server instance is functioning properly. Working state.
- AS\_Fail: All application server instances have failed. Failure state.
- HADB\_Fail: At least one pair of HADB nodes has a double node failure. Failure state.

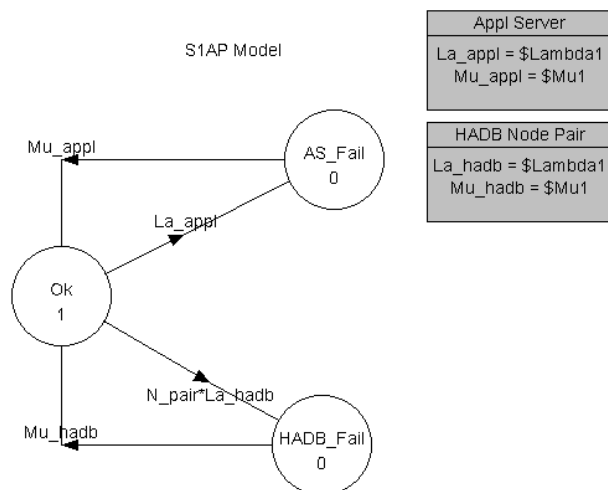


Figure 3. Sun Java System Application Server Enterprise Edition 7 System Model

Figure 3 shows that the system is normally working in the Ok state. It goes into state AS\_Fail at the rate  $La\_appl$  and comes back at the rate  $Mu\_appl$ , where  $La\_appl$  and  $Mu\_appl$  are the failure rate and recovery rate evaluated from the submodel “Appl Server.” The system also goes into state HADB\_Fail at the rate  $N\_pair * La\_hadb$  and comes back at the rate  $Mu\_hadb$ , where  $La\_hadb$  and  $Mu\_hadb$  are the failure rate

and recovery rate evaluated from the subcomponent model “HADB Node Pair,” and  $N\_pair$  is the number of HADB node pairs in the system.

Figure 4 shows the HADB Node Pair subdiagram. The state notation used in this diagram is:

- Ok: Both nodes are functioning. Working state.
- RestartShort: A node is being restarted from an HADB failure. Working state.
- RestartLong: A node is being restarted from an OS failure. Working state.
- Repair: A spare node is being rebuilt to replace a node with hardware failure. Working state.
- Maintenance: A node to be serviced is switching over to a standby node. Working state.
- 2\_Down: Both nodes are down. Failure state.

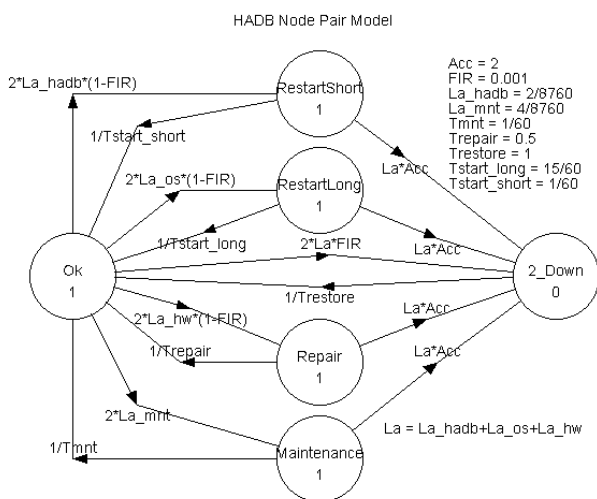


Figure 4. HADB Node Pair Model

Figure 4 clearly shows the transitions and associated rates at which the system moves between states. When a failure occurs with a probability of  $1-FIR$  (successful recovery), the system goes into a recovery state (RestartShort, RestartLong, or Repair), depending on the type of failure. The system also goes into failure state with probability  $FIR$  (unsuccessful recovery), which is a coefficient of the overall failure rate  $La$ . When the system is in recovery or maintenance state, only one node is functioning. As stated previously, the failure rate on the remaining node is assumed to be accelerated by a factor of two (2). A second failure on this node results in data loss and system failure. Should a system failure occur, it would take one hour (at the rate of  $1/Trestore$ ) for the system to restore to its working state.

Figure 5 is the Sun java System Application Server Enterprise Edition Model (two instances) subdiagram. The state notation is:

- All\_Work: All instances are functioning. Working state.
- Recovery: After an application server failure, sessions originally running on the failed application server are being reestablished on the remaining application server. Working state.

1DownShort: An instance is being restarted from an application server failure.

Working state.

1DownLong: An instance is being restarted from a hardware or OS failure.

Working state.

2\_Down: Both instances are down. Failure state.

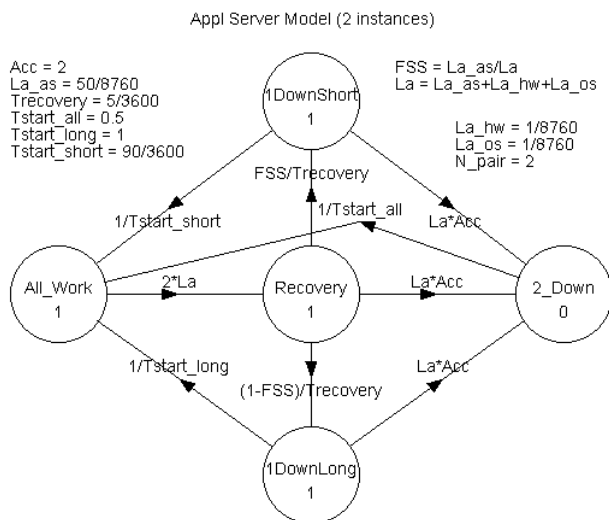


Figure 5. Sun Java System Application Server Enterprise Edition (Two Instances) Model

Figure 5 clearly shows the transitions and associated rates at which the system moves between states. The overall failure rate ( $La$ ) for an application server instance is the sum of the application server failure rate ( $La_{as}$ ), hardware failure rate ( $La_{hw}$ ), and OS failure rate ( $La_{os}$ ). When an instance fails, the system goes into recovery state and stays there for a short time interval ( $Trecovery$ ). Then with the probability of fraction of short start ( $FSS$ ) ( $FSS = La_{as}/La$ ), it will go into state 1DownShort, and stay there for a period of  $Tstart_{short}$ . Or with the probability of  $1-FSS$ , it will go into state 1DownLong, and stay there for a period of  $Tstart_{long}$  before going back to the normal state.

When the system is in states Recovery, 1DownShort, or 1DownLong, should a second failure (with an accelerated rate) occur on the remaining instance, the system goes into failure state 2\_Down, and stays there for  $Tstart_{all}$  (30 minutes) before going back to the normal state. But this is not the case when the configuration is four Sun Java System Application Server instances (Configuration 2), because such a configuration is able to tolerate up to three instance failures. The four-instance application server model, although used in the following analysis, is more complicated than this model and this paper does not discuss it in detail.

## 6. Analysis of Results

Given the models and parameters presented in previous sections, system results for the two modeled configurations generated by RAScad are shown in Table 3. For both Configuration 1 and Configuration 2, system availability is above five-nines. In Configuration 1, yearly downtime is dominated by the application server submodel (67 percent), while in Configuration 2, yearly downtime is dominated by the HADB submodel. When the number of application server instances is four or above, the application server submodel's yearly downtime is at the millisecond level, and can be ignored in availability analysis.

Table 3. System Results

Configuration	Availability	Yearly Downtime	Downtime Due to AS Submodel	Downtime Due to HADB Submodel
1	99.99933%	3.5 minutes	2.35 minutes (67%)	1.15 minutes (33%)
2	99.99956%	2.3 minutes	2.29 minutes (99.99%)	0.01 seconds (0.01%)

The model parameters can be classified into three categories: failure rate, recovery rate, and coverage (1-FIR) parameters. Failure rates cannot be accurately measured in limited time frames through testing and may vary on different customer sites, depending on the configuration, workloads, and environmental factors. All the failure rates used in the model in the uncertainty analysis, discussed later in this paper, are varied. FIR is estimated to be below 0.1 percent at the 95-percent confidence level, based on fault injection data. In the uncertainty analysis, it is allowed to go up to 0.2 percent, which is above the 99.5-percent confidence level upper bound.

Most recovery times (automatic restart time, repair time, and so on) are deterministic and are measured in the lab testing. An exception is the recovery time for hardware/OS failure on an application server node. This parameter is, to a large extent, controlled by customers. For example, a customer can minimize failure recovery times by implementing high-quality maintenance procedures and deploying a standby AS node.

The RAScad parametric analysis capability is employed to investigate how this parameter can impact availability. Figures 6 and 7 show the result of sensitivity analysis on hardware/OS failure recovery time (Tstart\_long varies from 30 minutes to three hours) for application server nodes. When the hardware/OS failure recovery time increases to 2.5 hours, the five-nines availability is no longer retained for

Configuration 1. However, even if the hardware/OS failure recovery time increases to three hours, 99.9995-percent availability is still retained for Configuration 2.

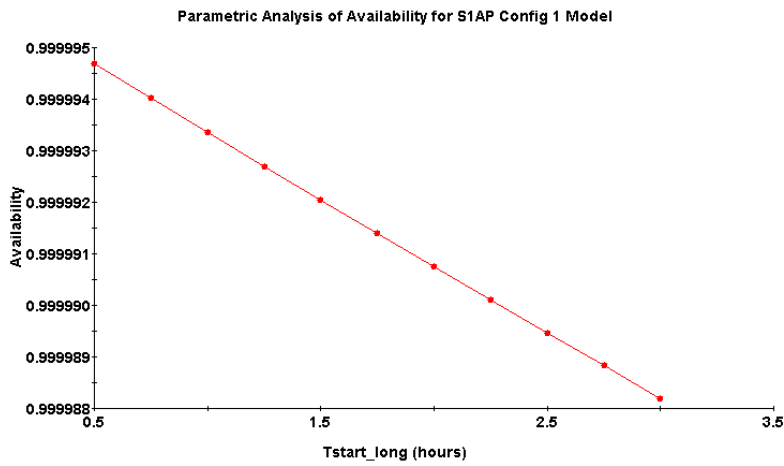


Figure 6. Sensitivity of Availability to Hardware/OS Recovery Time for Sun Java System Application Server Enterprise Edition Node – Configuration 1

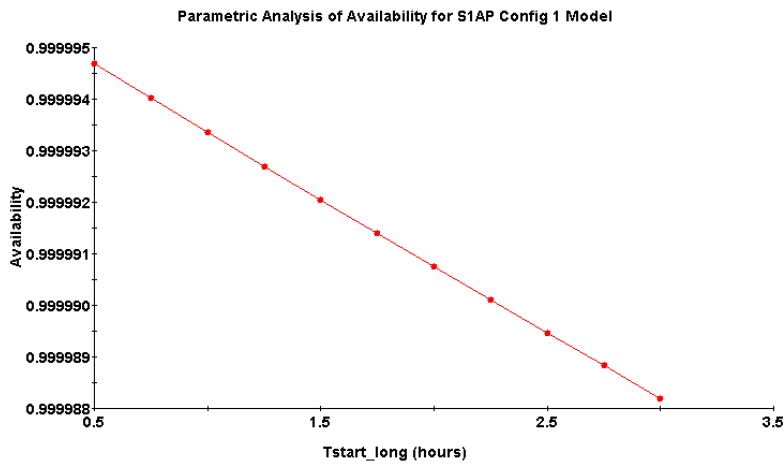


Figure 7. Sensitivity of Availability to Hardware/OS Recovery Time for Sun Java System Application Server Enterprise Edition Node – Configuration 2

One of the advanced analysis capabilities of RAScad is uncertainty analysis, which performs random sampling from parameter ranges defined by the user. This analysis method can address questions such as:

Assume there are N systems with each system’s parameters selected by random sampling from possible ranges in customer sites. What is the average system availability and confidence intervals?

In the uncertainty analysis for this study, we select six parameters which either cannot be accurately measured in limited time frames through testing, or may vary on different customer sites. The selected parameters and their varying ranges are:

- HADB failure rate — La\_hadb: 1/year – 4/year
- Application server failure rate — La\_as: 10/year – 50/year
- OS failure rate — La\_os: 0.5/year – 2/year
- Hardware failure rate — La\_hw: 0.5/year – 2/year
- Application server hardware/OS failure recovery time — Tstart\_long: 0.5 – 3 hours
- Fraction of imperfect recovery — FIR: 0 – 0.2 percent

The uncertainty analysis results for Configuration 1 and Configuration 2, generated for a sample size of 1,000 systems, are shown in Figures 8 and 9, respectively. For Configuration 1, the average yearly downtime for 1,000 systems is 3.82 minutes, and the 80-percent confidence interval is (1.95 min., 6.09 min.). Approximately 75 percent of sampled systems have yearly downtime of less than 5.25 minutes, or above 99.999-percent availability. For Configuration 2, the average yearly downtime is 2.9 minutes, and the 80-percent confidence interval is (1.04 min., 5.15 min.). More than 90 percent of sampled systems have yearly downtime of less than 5.25 minutes, or above 99.999-percent availability. Note the different scale on the vertical axis on Figures 8 and 9.

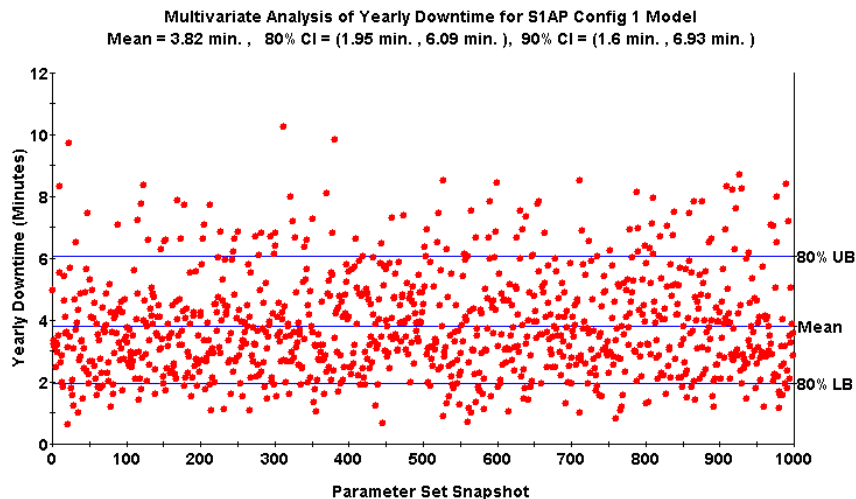


Figure 8. Uncertainty Analysis Result for Configuration 1

Finally, system availability is compared for various configurations listed in Table 4. One instance means there are no failover mechanisms and the server can be restarted in 90 seconds for application server failures, or in one hour for hardware/OS failures. The following observations are obtained from the results:

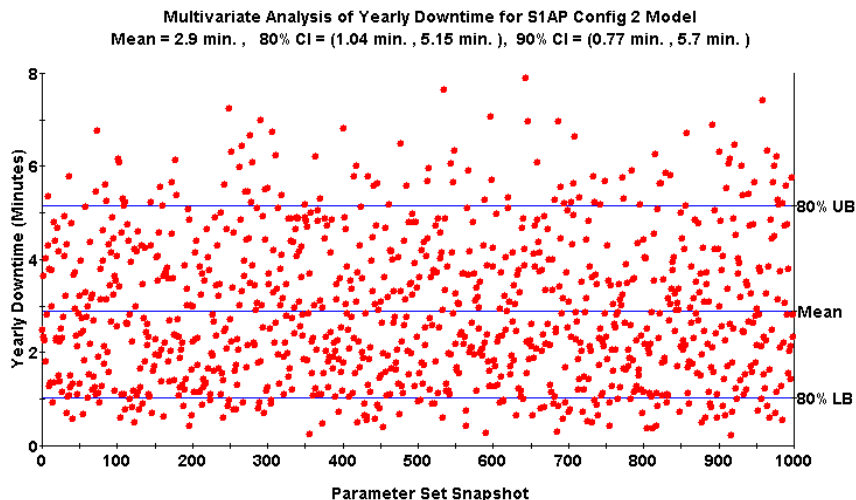


Figure 9. Uncertainty Analysis Result for Configuration 2

- Availability is significantly improved from a one-instance configuration to a two-instance configuration. That is, redundancy and failover provisions in the Sun Java System Application Server Enterprise Edition 7 product enhance system availability by two-nines.
- When the number of application server instances increases to four, availability is dominated by HADB. While increasing the number of HADB node pairs provides better scalability, it also introduces more chances to lose fragments of data, resulting in system failures. The 99.999-percent availability levels can no longer hold when the number of HADB node pairs reaches 10.
- The configuration with four application server instances and four HADB node pairs is the optimal configuration in terms of availability.

Table 4. Comparison of Configurations

Number of Instances	Number of HADB Pairs	Availability	Yearly Downtime	MTTF (Hours)
1	N/A	99.9629%	3.25 hours	168
2	2	99.99933%	3.3 minutes	89,980
4	4	99.99956%	2.3 minutes	229,285
6	6	99.99934%	3.43 minutes	152,871
8	8	99.99912%	4.58 minutes	114,659
10	10	99.99891%	5.73 minutes	91,730

## 7. Conclusions

This paper discusses an availability model and results generated from the model for the Sun Java System Application Server Enterprise Edition 7 software. The model is based on widely accepted methodology using the advanced RAS architecture modeling tool, RAScad. By employing conservative assumptions to build the model and estimate model parameters based on test and field data, system availability evaluates to above 99.999 percent. Uncertainty analysis on wide ranges of input parameters shows that average system availability is retained at the five-nines level. For Configuration 2, more than 90 percent of the sampled systems are above 99.999-percent availability. The analysis also shows that when the number of application server instances increases to four, system availability is dominated by HADB. The configuration of four application server instances and four pairs of HADB nodes is the optimal configuration in terms of availability.

Based on our test experience and the results of this analysis, we recommend the following configuration rules for achieving high levels of system availability:

- The system should be configured with built-in overcapacity to adequately handle load spikes as the result of node failures.
- To achieve high levels of availability, the system should have two to eight application server instances.
- The system should always have at least two spare HADB nodes.
- It is preferable to have one redundant cluster to facilitate application upgrades. This redundancy can be achieved by installing another application server on all participating nodes that host application server instances.
- Configure the application server and HADB to start automatically with OS booting.

## References

1. Hsueh, M. C., and Iyer, R. K. "Performability Modeling Based on Real Data: A Case Study," IEEE Transactions on Computers, April 1988, pp. 478-484.
2. Lee, I., Tang, D., Iyer, R. K., and Hsueh, M. C. "Measurement-Based Evaluation of Operating System Fault Tolerance," IEEE Transactions on Reliability, June 1993, pp. 238-249.
3. Tang, D., and Hecht, M. "Evaluation of Software Dependability Based on Stability Test Data," Proceedings of the 25th International Symposium on Fault-Tolerant Computing (FTCS-25), June 1995, pp. 434-443.
4. Kececioglu, D. Reliability and Life Testing Handbook, Vol. 1 & 2, PTR Prentice Hall, Englewood Cliffs, NJ, 1993.
5. Goyal, A., Lavenberg, S., and Trivedi, K. S. "Probabilistic Modeling of Computer System Availability," Annals of Operations Research, No. 8, March 1987, pp. 285-306.
6. Trivedi, K.S. Probability & Statistics with Reliability, Queuing, and Computer Science Applications, Second Edition, John Wiley & Sons, Inc., New York, 2002.
7. Sahner, R. A., and Trivedi, K. S. "Reliability Modeling Using SHARPE," IEEE Transactions on Reliability, Feb. 1987, pp. 186-193.
8. Tang, D., et al. "MEADep: A Dependability Evaluation Tool for Engineers," IEEE Transactions on Reliability, Dec. 1998, pp. 443-450.
9. Tang, D., Zhu, J., and Andrada, R. "Automatic Generation of Availability Models in RAScad," Proceedings of the International Conference on Dependable Systems and Networks (DSN-2002), June 2002, pp. 488-42.
10. Pramanick, I. "Modeling Sun Cluster Availability," Sun Users Performance Group Conference, SUPerG-2002, San Francisco, 2002.
11. Torbjornsen, Oystein. Multi-Site Declustering Strategies for Very High Database Service Availability, Dr. Ing. thesis, Division of Computer Science and Telematics, The Norwegian Institute of Technology, University of Trondheim, Norway, 1995.
12. For more information about SilkPerformer, visit [segue.com/html/s\\_solutions/s\\_performer/s\\_performer.htm](http://segue.com/html/s_solutions/s_performer/s_performer.htm)
13. Iyer, R. K., and Rossetti, D.J. "Effect of System Workload on Operating System Reliability: A Study on IBM 3081," IEEE Transactions on Software Engineering, Dec. 1985, pp. 1438-1448.

14. Brown, A., and Patterson, D. A. "To Err is Human," Proceedings of the First Workshop on Evaluating and Architecting System Dependability (EASY 01), Göteborg, Sweden, July 2001.

## More Information

For more information about the Sun Java System Application Server Enterprise Edition 7 software, please visit [sun.com/software/products/appsrvr/home\\_appsrvr.html](http://sun.com/software/products/appsrvr/home_appsrvr.html).

**SUN**<sup>TM</sup> © 2003 Sun Microsystems, Inc. All rights reserved.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

#### TRADEMARKS

Sun, Sun Microsystems, the Sun logo, Solaris, Sun, Sun Java System Application Server, Sun Java System Application Server Enterprise Edition, Sun Java System Directory Server, Java, J2EE, JavaServer Pages, JSP, JavaScript, Java Servlet, JDK, Sun Cluster, Solaris, Sun Enterprise, Sun Ultra, and JDBC are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc., in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Printed in USA 9/03

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-800-555-9786 or 1-800-555-9 SUN Web [sun.com](http://sun.com)



Sun Worldwide Sales Offices: Africa (North, West and Central) +33-13-067-4680, Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India-Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +822-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand-Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China- Beijing +86-10-6803-5588; Chengdu +86-28-619-9333, Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Singapore +65-6438-1888, Slovak Republic +421-2-4342-9485, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland-German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44 0 1252 420000, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905 3800