



Sun Java™ System

Application Server 7 System Deployment Guide

2004Q2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-5054

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF SUN MICROSYSTEMS, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF SUN MICROSYSTEMS, INC.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

Use is subject to license terms. This distribution may include materials developed by third parties.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, Sun™ ONE, Sun™ ONE Studio, iPlanet, J2EE, J2SE, Enterprise JavaBeans, EJB, JavaServer Pages, JSP, JDBC, JDK, JVM, Java Naming and Directory Interface, JavaMail, and the Java Coffee Cup logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

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

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

This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse <http://www.sun.com/patents> et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats-Unis et dans les autres pays.

CE PRODUIT CONTIENT DES INFORMATIONS CONFIDENTIELLES ET DES SECRETS COMMERCIAUX DE SUN MICROSYSTEMS, INC. SON UTILISATION, SA DIVULGATION ET SA REPRODUCTION SONT INTERDITES SANS L'AUTORISATION EXPRESSE, ECRITE ET PREALABLE DE SUN MICROSYSTEMS, INC.

L'utilisation est soumise aux termes de la Licence. Cette distribution peut comprendre des composants développés par des tierces parties.

Sun, Sun Microsystems, le logo Sun, Java, Solaris, Sun™ ONE, Sun™ ONE Studio, iPlanet, J2EE, J2SE, Enterprise JavaBeans, EJB, JavaServer Pages, JSP, JDBC, JDK, JVM, Java Naming and Directory Interface, JavaMail, et le logo Java Coffee Cup sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Ce produit est soumis à la législation américaine en matière de contrôle des exportations et peut être soumis à la réglementation en vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.

Contents

Who Should Use This Guide	5
Using the Documentation	6
How This Guide is Organized	8
Documentation Conventions	9
General Conventions	9
Conventions Referring to Directories	10
Contacting Sun	10
Give Us Feedback	11
Obtain Training	11
Contact Product Support	11
Chapter 1 Overview of Deployment	13
About Deployment	13
Throughput	14
Response Time	14
Availability	14
Phases of the Deployment Process	14
Planning Your Environment	15
Selecting a Topology	15
Running Tests	16
Understanding Session Persistence	16
Chapter 2 Planning your Environment	19
Introducing HADB	19
Active Nodes	20
Spare Nodes	21
Sample Spare Node Configuration 1	21

Sample HADB Architecture	22
Establishing Performance Goals	23
Estimating Throughput	23
Estimating Load on Application Server Instances	24
Calculating Maximum Number of Concurrent Users	24
Calculating Think Time	25
Calculating Average Response Time	25
Calculating Requests Per Minute	27
Estimating Load on the HADB	28
Number of Requests per Minute Received by the HADB	28
Session Size Per Request	29
Design Decisions to Make	32
Number of Application Server Instances Required	32
Number of HADB Nodes Required	32
HADB Storage Capacity Required	32
Designing for Peak Load Compared to Steady State Load	34
Planning Network Configuration to Meet Your Performance Goals	35
Estimating Bandwidth Requirements	35
Calculating Bandwidth Required	36
Estimating Peak Load	37
Configuring Subnets	37
Choosing Network Cards	38
Network Settings for HADB	38
Planning for Availability	38
Adding Redundancy to the System	39
Identifying Failure Classes	39
Using Redundancy Units to Improve Availability	39
Using Spare Nodes to Improve Fault Tolerance	40
Planning Failover Capacity	40
Using Multiple Clusters to Improve Availability	40
 Chapter 3 Selecting a Topology	 41
Common Requirements	42
General Requirements	42
HADB Nodes and Machines	43
Load Balancer Configuration	43
Co-located Topology	44
Reference Co-located Topology	44
Configuration Settings for Reference Co-located Topology	46
Variation to Reference Co-located Topology	47
Configuration Settings for Variation to the Reference Co-located Topology	50
Separate Tier Topology	51
Reference Configuration	51

Configuration Settings for Reference Separate Tier Topology	53
Variation to Reference Separate Tier Topology	54
Configuration Settings for Variation to Reference Separate Tier Topology	56
Comparison of Topologies	57
Determining Which Topology to Use	58
Appendix A Checklist for Deployment	59
Index	67

About This Guide

This guide describes how to create and run Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications that follow the new open Java standards model for Java™ Servlet, JavaServer Pages™ (JSP™), Enterprise JavaBeans™ (EJB™), and other J2EE components in the Sun Java™ System Application Server 7 Enterprise Edition environment.

This preface addresses the following topics:

- [Who Should Use This Guide](#)
- [Using the Documentation](#)
- [How This Guide is Organized](#)
- [Documentation Conventions](#)
- [Contacting Sun](#)

Who Should Use This Guide

The information in this guide is intended for system administrators who want to deploy and support a large system that has high availability requirements.

This guide assumes you are familiar with the following:

- Installation of enterprise-level software products
- UNIX® operating system
- Client/server programming model
- Internet and World Wide Web
- Availability and clustering concepts

Using the Documentation

The Sun Java System Application Server Standard and Enterprise Edition manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML).

The following table lists tasks and concepts described in the Sun Java System Application Server manuals. The manuals marked *(updated for 2004Q2)* have been updated for the Sun Java System Application Server 7 Standard and Enterprise Edition release. The manuals not marked in this way have not been updated since the version 7.0 Enterprise Edition release.

Table 1 Sun Java System Application Server Documentation Roadmap

For information about	See the following
<i>(Updated for 7 2004Q2)</i> Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of supported hardware, operating system, JDK, and JDBC/RDBMS.	<i>Release Notes</i>
Sun Java System Application Server 7 overview, including the features available with each product edition.	<i>Product Overview</i>
Diagrams and descriptions of server architecture and the benefits of the Sun Java System Application Server architectural approach.	<i>Server Architecture</i>
<i>(Updated for 7 2004Q2)</i> How to get started with the Sun Java System Application Server product. Includes a sample application tutorial. There are two guides, one for Standard Edition and one for Enterprise Edition.	<i>Getting Started Guide</i>
<i>(Updated for 7 2004Q2)</i> Installing the Sun Java System Application Server Standard Edition and Enterprise Edition software and its components, such as sample applications and the Administration interface. For the Enterprise Edition software, instructions are provided for implementing the high-availability configuration.	<i>Installation Guide</i>
<i>(Updated for 7 2004Q2)</i> Evaluating your system needs and enterprise to ensure that you deploy Sun Java System Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying an application server are also discussed.	<i>System Deployment Guide</i>
Creating and implementing Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Sun Java System Application Server that follow the open Java standards model for J2EE components such as servlets, Enterprise JavaBeans™ (EJBs™), and JavaServer Pages™ (JSPs™). Includes general information about application design, developer tools, security, assembly, deployment, debugging, and creating lifecycle modules. A comprehensive Sun Java System Application Server glossary is included.	<i>Developer's Guide</i>

Table 1 Sun Java System Application Server Documentation Roadmap (*Continued*)

For information about	See the following
(Updated for 7 2004Q2) Creating and implementing J2EE web applications that follow the Java™ Servlet and JavaServer Pages (JSP) specifications on the Sun Java System Application Server. Discusses web application programming concepts and tasks, and provides sample code, implementation tips, and reference material. Topics include results caching, JSP precompilation, session management, security, deployment, SHTML, and CGI.	<i>Developer's Guide to Web Applications</i>
(Updated for 7 2004Q2) Creating and implementing J2EE applications that follow the open Java standards model for enterprise beans on the Sun Java System Application Server. Discusses Enterprise JavaBeans (EJB) programming concepts and tasks, and provides sample code, implementation tips, and reference material. Topics include container-managed persistence, read-only beans, and the XML and DTD files associated with enterprise beans.	<i>Developer's Guide to Enterprise JavaBeans Technology</i>
(Updated for 7 2004Q2) Creating Application Client Container (ACC) clients that access J2EE applications on the Sun Java System Application Server.	<i>Developer's Guide to Clients</i>
Creating web services in the Sun Java System Application Server environment.	<i>Developer's Guide to Web Services</i>
(Updated for 7 2004Q2) Java™ Database Connectivity (JDBC™), transaction, Java Naming and Directory Interface™ (JNDI), Java™ Message Service (JMS), and JavaMail™ APIs.	<i>Developer's Guide to J2EE Services and APIs</i>
Creating custom NSAPI plug-ins.	<i>Developer's Guide to NSAPI</i>
(Updated for 7 2004Q2) Information and instructions on the configuration, management, and deployment of the Sun Java System Application Server subsystems and components, from both the Administration interface and the command-line interface. Topics include cluster management, the high-availability database, load balancing, and session persistence. A comprehensive Sun Java System Application Server glossary is included.	<i>Administration Guide</i>
(Updated for 7 2004Q2) Editing Sun Java System Application Server configuration files, such as the <code>server.xml</code> file.	<i>Administrator's Configuration File Reference</i>
Configuring and administering security for the Sun Java System Application Server operational environment. Includes information on general security, certificates, and SSL/TLS encryption. HTTP server-based security is also addressed.	<i>Administrator's Guide to Security</i>
Configuring and administering service provider implementation for J2EE™ Connector Architecture (CA) connectors for the Sun Java System Application Server. Topics include the Administration Tool, Pooling Monitor, deploying a JCA connector, and sample connectors and sample applications.	<i>J2EE CA Service Provider Implementation Administrator's Guide</i>
(Updated for 7 2004Q2) Migrating your applications to the new Sun Java System Application Server programming model, specifically from iPlanet Application Server 6.x and Sun ONE Application Server 7.0. Includes a sample migration.	<i>Migrating and Redeploying Server Applications Guide</i>
(Updated for 7 2004Q2) How and why to tune your Sun Java System Application Server to improve performance.	<i>Performance Tuning Guide</i>

Table 1 Sun Java System Application Server Documentation Roadmap (*Continued*)

For information about	See the following
(Updated for 7 2004Q2) Information on solving Sun Java System Application Server problems.	<i>Troubleshooting Guide</i>
(Updated for 7 2004Q2) Information on solving Sun Java System Application Server error messages.	<i>Error Message Reference</i>
(Updated for 7 2004Q2) Utility commands available with the Sun Java System Application Server; written in manpage style.	<i>Utility Reference Manual</i>
Using the Sun™ Java System Message Queue 3.5 software.	The Sun Java System Message Queue documentation at: http://docs.sun.com/db?p=prod/s1.slmsgqu
For information about	See the following
(Updated for 7 2004Q2) Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of supported hardware, operating system, JDK, and JDBC/RDBMS.	<i>Release Notes</i>

How This Guide is Organized

This guide is divided into three chapters. Read the chapters in the order they are presented as each chapter builds on the previous one.

- [Chapter 1, “Overview of Deployment”](#) provides an introduction to deployment and discusses phases of deployment.
- [Chapter 2, “Planning your Environment”](#) discusses the steps you need to determine the environment that best suits your business needs.
- [Chapter 3, “Selecting a Topology”](#) contains examples of application server topologies, and helps you determine the topology that best suits your business needs.
- [Appendix A, “Checklist for Deployment”](#) provides a list of tasks to get started with Sun Java System Application Server.

Documentation Conventions

This section describes the types of conventions used throughout this guide:

- [General Conventions](#)
- [Conventions Referring to Directories](#)

General Conventions

The following general conventions are used in this guide:

- **File and directory paths** are given in UNIX® format (with forward slashes separating directory names). For Windows versions, the directory paths are the same, except that backslashes are used to separate directories.

- **URLs** are given in the format:

`http://server.domain/path/file.html`

In these URLs, *server* is the server name where applications are run; *domain* is your Internet domain name; *path* is the server's directory structure; and *file* is an individual filename. Italic items in URLs are placeholders.

- **Font conventions** include:
 - The `monospace` font is used for sample code and code listings, API and language elements (such as function names and class names), file names, pathnames, directory names, and HTML tags.
 - *Italic* type is used for code variables.
 - *Italic* type is also used for book titles, emphasis, variables and placeholders, and words used in the literal sense.
 - **Bold** type is used as either a paragraph lead-in or to indicate words used in the literal sense.
- **Installation root directories** for most platforms are indicated by *install_dir* in this document. Exceptions are noted in [“Conventions Referring to Directories” on page 10](#).

By default, the location of *install_dir* on **most** platforms is:

- Solaris 8 and 9 and Linux file-based installations:

user's home directory/sun/appserver7

- Windows, all installations:

`C:\Sun\AppServer7`

For the platforms listed above, *default_config_dir* and *install_config_dir* are identical to *install_dir*. See “[Conventions Referring to Directories](#)” on page 10. for exceptions and additional information.

- **Instance root directories** are indicated by *instance_dir* in this document, which is an abbreviation for the following:

default_config_dir/domains/domain/instance

- **UNIX-specific descriptions** throughout this manual apply to the Linux operating system as well, except where Linux is specifically mentioned.

Conventions Referring to Directories

By default, when using the Solaris 8 and 9 package-based or Linux RPM-based installation, the application server files are spread across several root directories. This guide uses the following document conventions to correspond to the various default installation directories provided:

- *install_dir* refers to `/opt/SUNWappserver7`, which contains the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.
- *default_config_dir* refers to `/var/opt/SUNWappserver7/domains`, which is the default location for any domains that are created.
- *install_config_dir* refers to `/etc/opt/SUNWappserver7/config`, which contains installation-wide configuration information such as licenses and the master list of administrative domains configured for this installation.

Contacting Sun

You might want to contact Sun Microsystems in order to:

- [Give Us Feedback](#)
- [Obtain Training](#)
- [Contact Product Support](#)

Give Us Feedback

If you have general feedback on the product or documentation, please send this to:

<http://www.sun.com/hwdocs/feedback/>

Obtain Training

Application Server training courses are available at:

http://training.sun.com/US/catalog/enterprise/web_application.html/

Visit this site often for new course availability on the Sun Java System Application Server.

Contact Product Support

If you have problems with your system, contact customer support using one of the following mechanisms:

- The online support web site at:
<http://www.sun.com/supporttraining/>
- The telephone dispatch number associated with your maintenance contract

Please have the following information available prior to contacting support. This helps to ensure that our support staff can best assist you in resolving problems:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem. Here are some of the commonly used commands:
 - **Solaris:** pkginfo, showrev
 - **Linux:** rpm
 - **All:** asadmin version --verbose
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps
- Configuration files such as:

- *instance_dir*/config/server.xml
- a web application's web.xml file,
when a web application is involved in the problem
- For an application, whether the problem appears when it is running in a cluster or standalone

Overview of Deployment

This chapter describes what you need to know to set up your Sun Java™ System Application Server, Standard and Enterprise Edition the way that best meets your requirements.

This chapter contains the following sections:

- [About Deployment](#)
- [Phases of the Deployment Process](#)

About Deployment

Successful deployment of complex applications on the Application Server requires that you consider practical aspects of the environment. In general, you should begin by assessing your goals for performance and availability. You should then plan the hardware, network, and resource configuration accordingly.

Here are the important goals that you should consider while planning the deployment:

- Throughput
- Response time
- Availability

You gather information related to these goals and then analyze it to establish a processing threshold for your site.

In considering performance, consider both—application server instances and the High Availability Database (HADB).

The HADB uses the patented Always-On technology and works as the persistence store to provide high availability for web applications. It offers an ideal platform for delivering all types of session state persistence within an enterprise application server environment. For more information on configuring HADB, see *Sun Java™ System Application Server Administration Guide*.

Throughput

Throughput is the number of requests that Sun Java System Application Server can service in a given time period. For example requests per minute. You should estimate the maximum number of operations and transactions that the system needs to perform under peak load conditions. It is also useful to determine the operations and transactions per minute under steady state (typical) load conditions. This will help you to determine the network bandwidth needed, the number of application server instances required, and the number of HADB nodes required.

You should also consider plans to increase capacity in the future.

Response Time

You should determine the acceptable response time from the system under heavy load. This has a direct bearing on hardware capacity planning.

Availability

Will your system be running 24 x 7? If there is a failure in the system, will your users notice it? Do you have a subset of applications that need to be available all the time whereas other applications will run only periodically? The answers to these questions determine your availability needs. You will have to build redundancy into the system to meet availability needs and avoid single points of failure.

Phases of the Deployment Process

The deployment process primarily comprises of the following three phases, each one building on the previous one.

- [Planning Your Environment](#)

- [Selecting a Topology](#)
- [Running Tests](#)

Planning Your Environment

In the first phase of planning your deployment, determine how the Application Server fits into your overall enterprise. Central to planning your environment is the assessment of the goals discussed in [“About Deployment” on page 13](#). You should establish performance goals related to throughput and response time. You also determine your availability goals.

Based on the performance and availability goals, you consider the network requirements and the infrastructure requirements including hardware, storage, and network requirements.

You may realize during this process that you should change the structure and components of your network to accommodate the needs of your Application Server. If your network structure cannot be changed at this time, use the environment planning process to determine how you can best deploy Application Server to fit in with your existing network.

For more details on this phase of the planning process, see [Chapter 2, “Planning your Environment.”](#)

Selecting a Topology

Once you have determined the performance, availability, network, and infrastructure requirements, you then select a topology that best meets your performance needs. A topology is the schematic arrangement of Application Server components and the communication flow between these components. The two recommended topologies (and their variations) are:

- **Co-located:** The application server instance and the HADB node are on the same machine.
- **Separate Tier:** The application server instance and the HADB node are on different machines.

For more details on these topologies, see [Chapter 3, “Selecting a Topology.”](#)

Running Tests

Once you configure the Application Server, you should deploy a representative sampling of applications and run tests to check whether your performance goals are met. If you are not able to reach your stated performance goals, use this phase to identify bottlenecks, fine-tune the system, and improve performance.

As implementation of this phase is completely dependent on your particular environment, it is not covered in this guide.

Understanding Session Persistence

As an application session proceeds, there is often data that is part of the session that is not stored in a traditional database. An example of such data is the content of your shopping cart. Sun Java System Application Server provides the capability to save, or persist, this session state in a repository, so that if an application server instance experiences a failure, the session state can be recovered and the session can continue without loss of information.

Sun Java System Application Server supports persistence of HTTP sessions and stateful session bean (SFSB) sessions. The persistence types `ha`, `memory`, and `file` are supported by the Sun Java System Application Server. When you set the persistence type to `ha`, the HADB is used as the persistence store for both HTTP and SFSB sessions. If the persistence type is *not* set to `ha`, SFSB sessions are persisted to the session store defined for passivated SFSBs. The `memory` and `file` settings have no effect on SFSBs.

When an SFSB's state is checked for changes that need to be saved, this is called checkpointing. If SFSB checkpointing is enabled, checkpointing generally occurs after any transaction involving the SFSB is completed, even if the transaction rolls back.

For more information on enabling SFSB checkpointing, see the *Sun Java System Application Server Developer's Guide to Enterprise JavaBeans Technology*.

Apart from the number of requests being served by the Application Server, the session persistence configuration settings affect the number of requests received per minute by the HADB, as well as the session information in each request.

The persistence settings can be defined for each application server instance. However, all application server instances in a single cluster must have the same persistence configuration. If you have deployed more than one Application Server cluster, it is not necessary for all clusters to have the same persistence configuration settings.

For more information on configuring session persistence and its effect on performance, see “[Comparison of Persistence Scope Options](#)” on page 30.

NOTE	Use the command <code>cladmin</code> to ensure that the session persistence settings are homogeneous for all instances in a cluster. For more information on using the <code>cladmin</code> command, see <i>Sun Java System Application Server Administration Guide</i> .
-------------	---

Planning your Environment

Planning your environment is one of the first phases of deployment. In this phase, you should first decide your performance and availability goals, and then accordingly make decisions about the hardware, network, and storage requirements.

The main objective of this phase is to determine the environment that best meets your business requirements.

This chapter contains the following sections:

- [Introducing HADB](#)
- [Establishing Performance Goals](#)
- [Design Decisions to Make](#)
- [Planning Network Configuration to Meet Your Performance Goals](#)
- [Planning for Availability](#)

Introducing HADB

This section contains the following topics:

- [Active Nodes](#)
- [Spare Nodes](#)
- [Sample HADB Architecture](#)

Sun Java™ System Application Server 7 Enterprise Edition supports persistence of HTTP sessions, Stateful Session Beans, and remote references of EJB look-ups on the RMI/IIOP path. The high-availability database (HADB), bundled with the Enterprise Edition of Application Server provides a highly available persistence store.

An HADB system is comprised of various *nodes*. Each HADB node consists of the following:

- a set of processes
- a dedicated area of shared memory
- one or more secondary storage devices

It is used for storing and updating session data.

There are two types of HADB nodes:

- [Active Nodes](#)

An active node is a node that stores data.

- [Spare Nodes](#)

A spare node does not contain any data initially, but can start performing as an active node if an active node becomes unavailable.

HADB nodes are organized into two Data Redundancy Units (DRUs) that mirror each other. Each DRU consists of half of the active node, and half of the spare node. One DRU contains one complete copy of the data.

To ensure fault tolerance, the servers that support one DRU must be completely self-supported with respect to power (use of uninterruptible power supplies is needed), processing units, and storage. If a power failure occurs in one DRU, the nodes in the other DRU can continue servicing requests until the power returns.

NOTE Machines that host HADB Nodes must be added in pairs, with one machine in each DRU.

Active Nodes

Each active node must have a mirror node; that is active nodes must be configured in pairs. In addition, to maximize HADB availability, you should include two spare nodes for each pair so that if an active node fails, a spare node can take over while the failed node is repaired.

Spare Nodes

A spare node is an additional HADB node connected to a DRU. A spare node initially does not contain data, but constantly monitors for failure of active nodes in the DRU. If an active node fails, the spare node takes over the functions of the failed node while the failed node is being repaired.

Though you can configure an HADB system without spare nodes, it is not recommended. If the machine running an active node fails, the other nodes (including the mirror node) will become overloaded and drastically reduce performance. Depending on the impact of losing one machine, this may make your system effectively unavailable because machines running the other nodes also become overloaded.

Moreover, your system will be running without fault tolerance till you repair the failed machine as there is no mirror node to replicate the data. For high availability, you should minimize the time during which the system functions with only a single node.

Spare nodes are not mandatory, but should be used if you require high availability. Spare nodes allow a single machine to fail and yet maintain overall level of service. You should allocate one machine for each DRU to act as a spare machine, so that if one of your machines fails, your HADB system can continue without adversely affecting performance. A spare node also makes it easy for you to perform planned maintenance on the machines that host the active nodes.

NOTE As a general rule, you should have a spare machine with enough Application Server instances and HADB nodes to replace any machine that becomes unavailable.

Sample Spare Node Configuration 1

If you have a co-located deployment, with four Sun Fire™ V480 servers where each server has one Application Server instance and two HADB data nodes, you should allocate two more servers as spare machines (one machine per DRU). Each spare machine should run one application server instance and two spare HADB nodes.

Sample Spare Node Configuration 2

Suppose you have a separate-tier deployment where the HADB tier has two Sun Fire™ 280R servers, each running two HADB data nodes. To maintain this system at full capacity, even if one machine becomes unavailable, configure one spare machine for the Application Server instances tier and one spare machine for the HADB tier.

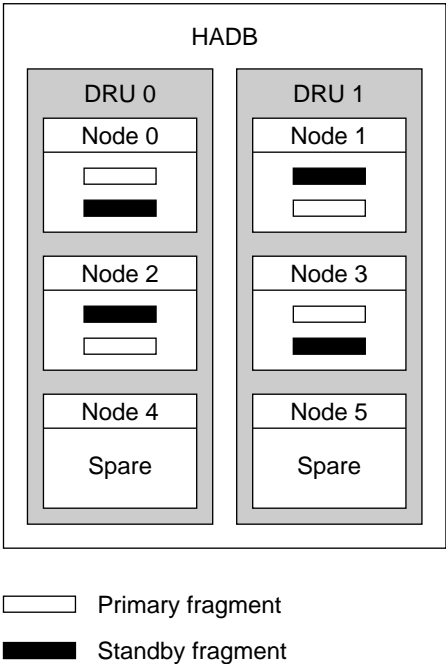
The spare machine for the Application Server instances tier should have as many instances as the other machines in the Application Server instances tier. Similarly, the spare machine for the HADB tier should have as many HADB nodes as the other machines in the HADB tier.

For more information about the co-located and the separate tier deployment topologies, see [Chapter 3, “Selecting a Topology.”](#)

Sample HADB Architecture

The following figure shows the architecture of a database with four active nodes and two spare nodes. Nodes 0 and 1 are a mirror node pair, as are nodes 2 and 3.

Figure 2-1 HADB Architecture with Four Active Nodes and Two Spare Nodes



Establishing Performance Goals

As explained in [Chapter 1, “Overview of Deployment,”](#) one of your main goals is to maximize performance. This essentially translates into maximizing throughput and reducing response time.

Beyond these basic goals, you should establish specific goals by determining the following:

- What capacity of requests, or throughput, can the system support?
- How many concurrent users can the system support?
- What is an acceptable average response time for requests submitted by your users?
- What is the average think time between requests?

These factors are interrelated. If you know the answer to any three of these four factors, you can calculate the fourth.

Some of the metrics described in this chapter can be calculated using a remote browser emulator (RBE) tool, or web site performance and benchmarking software, that simulates your enterprise’s web application activity. Typically, RBE and benchmarking products generate concurrent HTTP requests and then report back the response time and number of requests per minute. You can then use these figures to calculate server activity.

The results of the calculations described in this chapter are not absolute. Treat them as reference points to work against, as you fine-tune the performance of Sun Java System Application Server.

This section describes the following topics:

- [Estimating Throughput](#)
- [Estimating Load on Application Server Instances](#)
- [Estimating Load on the HADB](#)
- [Estimating Bandwidth Requirements](#)
- [Estimating Peak Load](#)

Estimating Throughput

Throughput, as measured for application server instances and for the HADB, has different implications.

A good measure of the throughput for Application Server instances is the number of requests precessed per minute. A good measure of throughput for the HADB is the number of requests precessed per minute by the HADB, and the session size per request. The session size per request is important because the size of session data stored varies from request to request.

For more information session persistence, see [Chapter 1, “Overview of Deployment.”](#)

Estimating Load on Application Server Instances

Consider the following factors to estimate the load on application server instances:

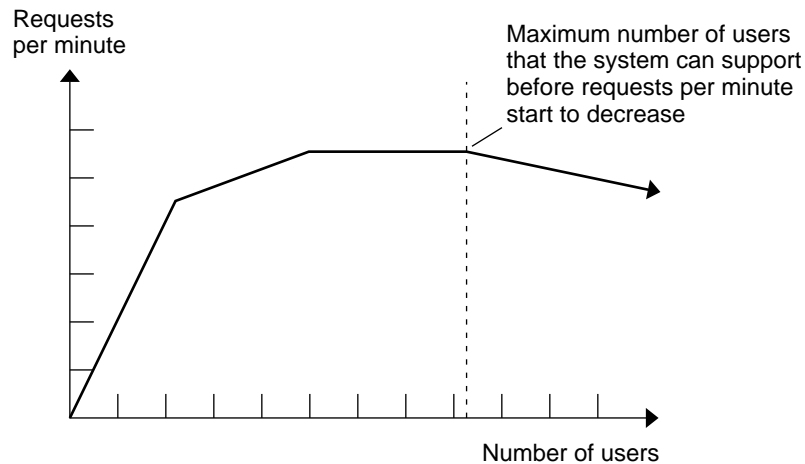
- [Calculating Maximum Number of Concurrent Users](#)
- [Calculating Think Time](#)
- [Calculating Average Response Time](#)
- [Calculating Requests Per Minute](#)

Calculating Maximum Number of Concurrent Users

A user runs a process (for example through a web-browser) that periodically sends requests from a client machine to the Application Server. When estimating the number of concurrent users, include all users currently active. A user is considered active as long as the session that user is running is active (for example, the session has neither expired nor terminated).

A user is concurrent for as long as the user is on the system as a running process submitting requests, receiving results of requests from the server, and viewing the results.

Eventually, as the number of concurrent users submitting requests increases, requests processed per minute begins to decline (and the response time begins to increase). The following diagram illustrates this situation.

Figure 2-2 Performance Pattern with Increasing Number of Users.

You should identify the point at which adding more concurrent users reduces the number of requests that can be processed per minute. This point indicates when performance starts to degrade.

Calculating Think Time

A user does not submit requests continuously. A user submits a request, the server receives the request, processes it and then returns a result, at which point the user spends some time analyzing the result before submitting a new request. The time spent reviewing the result of a request is called *think time*.

Determining the typical duration of think time is important. You can use the duration to calculate more accurately the number of requests per minute, as well as the number of concurrent users your system can support. Essentially, when a user is on the system but not submitting a request, a gap opens for another user to submit a request without altering system load. This implies that you can support more concurrent users.

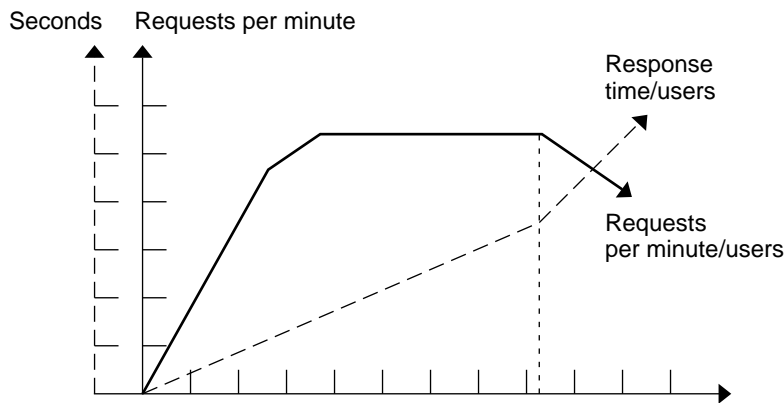
Calculating Average Response Time

Response time refers to the amount of time it takes for results of a request to be returned to the user. The response time is affected by a number of factors including network bandwidth, number of users, number and type of requests submitted, and average think time.

In this section, response time refers to the mean, or average, response time. Each type of request has its own minimal response time. However, when evaluating system performance, you should base your analysis on the average response time of all requests.

The faster the response time, the more requests per minute are being processed. However, as the number of users on your system increases, response time starts to increase as well, even though the number of requests per minute declines, as the following diagram illustrates:

Figure 2-3 Response Time with Increasing Number of Users



A system performance graph similar to [Figure 2-3 on page 26](#), indicates that after a certain point (point A in this diagram), requests per minute are inversely proportional to response time- the sharper the decline in requests per minute, the steeper the increase in response time (represented by the dotted line arrow).

In [Figure 2-3 on page 26](#), point A represents peak load, that is, the point at which requests per minute start to decline. Prior to this point response time calculations are not necessarily accurate because they do not use peak numbers in the formula. After this point, (because of the inversely proportional relationship between requests per minute and response time), you can more accurately calculate response time using maximum number of users and requests per minute.

To determine response time at peak load, use the following formula:

$$\text{Response time} = (\text{concurrent users} / \text{requests per second}) - \text{think time in seconds}$$

To obtain an accurate response time result, you must always include think time in the equation.

Example Calculation of Response Time

For example, if the following conditions exist:

- Maximum number of concurrent users that your system can support at peak load is 5,000.
- Maximum number of requests the system can process at peak load is 1,000 per second.
- Average think time equals 3 seconds per request.

$\text{Response time} = (5000 / 1000) - 3 \text{ seconds think time}$

Therefore, the response time is 2 seconds.

After you have calculated your system's response time, particularly at peak load, decide what is an acceptable response time for your enterprise. Response time, along with throughput, is one of the main factors critical to Sun Java System Application Server performance. Improving the response time should be one of your goals.

If there is a response time beyond which you do not want to wait, and performance is such that you get response times over that level, then work towards improving your response time or redefine your response time threshold.

Calculating Requests Per Minute

If you know the number of concurrent users at any given time, the response time of their requests and the average user think time at that time, you can determine requests per minute. Typically, you start by knowing how many concurrent users are on your system.

For example, after running a few web site performance calculation software, you conclude that the average number of concurrent users submitting requests on your online banking web site is 3,000. This number is dependent on the number of users who have signed up to be members of your online bank, their banking transaction behavior, the times of the day or week they choose to submit requests, and so on.

Therefore, knowing this information enables you to use the requests per minute formula described in this section to calculate how many requests per minute your system can handle for this user base. Since requests per minute and response time become inversely proportional at peak load, decide if fewer requests per minute are acceptable as a trade-off for better response time, or alternatively, if a slower response time is acceptable as a trade-off for more requests per minute.

Essentially, you should experiment with the requests per minute and response time thresholds that is acceptable as a starting point for fine-tuning system performance. Thereafter, decide which areas of your system you want to adjust.

The formula for obtaining the requests per second is as follows:

$$\text{requests per second} = \text{concurrent users} / (\text{response time in seconds} + \text{think time in seconds})$$

Example Calculation of Requests per Second

For example, if the following conditions exists:

- Concurrent users equals 2,800.
- Average response time equals 1 second per request.
- Average think time equals 3 seconds.

$$\text{Requests per second} = 2800 / (1+3)$$

Therefore, the number of requests per second is 700 and the number of requests per minute is 42000.

Estimating Load on the HADB

To calculate load on the HADB, consider the following factors:

- [Number of Requests per Minute Received by the HADB](#)
- [Session Size Per Request](#)

The session persistence settings that you specify affect the load on the HADB. For more information on configuring session persistence, see *Sun Java System Application Server Administration Guide*.

Number of Requests per Minute Received by the HADB

The number of requests per minute received by the HADB depends on the `persistence frequency`. This is the frequency at which HTTP session and SFSB session information is stored in the HADB, defined through the persistence frequency settings.

The persistence frequency options are:

- `web-method`: In the web-method persistence frequency mode, the session is stored after every web request.

The `web-method` persistency frequency provides the highest guarantee that the session information stored will be up to date. However, this option increases traffic to the HADB due to information being stored to the HADB at the end of every web request.

- `time-based`: In the time-based persistence frequency, the session is stored at configurable time intervals.

The `time-based` frequency reduces the traffic to the HADB but it provides less guarantee that the session information will be up to date, compared to the `web-method`.

This section discusses the following topics:

- [Comparison of Persistence Frequency Options](#)
- [Comparison of Persistence Scope Options](#)

Comparison of Persistence Frequency Options

Table 2-1 summarizes the advantages and disadvantages of both HTTP and SFSB session persistence frequency options. The left column lists the persistence frequency options, the middle column lists the advantages and the right column lists the disadvantages of using this option.

Table 2-1 Comparison of Persistence Frequency Options

Persistence Frequency Option	Advantage(s)	Disadvantage(s)
<code>web-method</code>	Guarantees that the most up-to-date session information is available.	Potentially increased response time and reduced throughput.
<code>time-based</code>	Better response time and potentially better throughput.	Less guarantee than the <code>web-method</code> persistence frequency that the most updated session information is available after the failure of an application server instance.

Session Size Per Request

The session size per request depends on the amount of session information stored in the session.

TIP	To improve overall performance, reduce the amount of information in the session as much as possible.
------------	--

You can further fine-tune the session size per request through the persistence scope settings. Choose from the following options for HTTP and SFSB session persistence scope:

- `session`: The entire session is saved every time session information is saved to the HADB database.
- `modified-session`: The session is saved only if it has been modified.
- `modified-attribute`: Only those attributes are stored that have been modified (inserted, updated, or deleted) since the last time the session was stored.

NOTE The persistence scope `modified-attribute` is not certified as a full production-quality feature. If you use this persistence scope, evaluate the performance and stability of Sun Java System Application Server in expected peak load conditions. If exceptions are logged or the response time is more than that is acceptable, do not use this persistence scope for your production environment.

Comparison of Persistence Scope Options

Table 2-2 summarizes the advantages and disadvantages of the persistence scope options. The left column lists the persistence frequency options, the middle column lists the advantages, and the right column lists the disadvantages of this option.

Table 2-2 Comparison of Persistence Scope Options

Persistence Scope Option	Advantage(s)	Disadvantage(s)
<code>modified-session</code>	Provides improved response time for requests that do not modify session state.	Your application must call the <code>setAttribute</code> method (if the attribute was changed) or the <code>removeAttribute</code> method (if the attribute was removed) on the session during the execution of a web method (typically <code>doGet</code> or <code>doPost</code>).
<code>session</code>	No constraint on applications.	Potentially poorer throughput and response time as compared to the <code>modified-session</code> and the <code>modified-attribute</code> options.

Table 2-2 Comparison of Persistence Scope Options (*Continued*)

Persistence Scope Option	Advantage(s)	Disadvantage(s)
modified-attribute	Better throughput and response time for requests in which the percentage of session state modified is low.	<ol style="list-style-type: none"> 1. As the percentage of session state that gets modified for a given request grows to around 60%, the throughput and the response time degrade. In such cases, the performance gets worse than the session or modified-session persistence scope because of the overhead of splitting the attributes into separate records. 2. Your application must be written to meet the following constraints required by this mode: <ul style="list-style-type: none"> • Call <code>setAttribute</code> or <code>removeAttribute</code> every time you modify the session state. • Make sure there are no cross references between attributes. • Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.

In the case of an SFSB session persistence, the load on HADB database depends on the number of SFSB beans enabled for checkpointing. Checkpointing generally occurs after any transaction involving the SFSB is completed (even if the transaction rolls back), and how many methods in each bean are enabled for checkpointing in the case of non-transactional methods.

For better performance, specify a small subset of methods for checkpointing. The size of the data that is being checkpointed and the frequency at which checkpointing takes place determines the additional overhead in response time for a given client interaction.

Design Decisions to Make

Depending on the load on the application server instances, the load on the HADB, and the failover requirements, you should make the following decisions at this stage:

- [Number of Application Server Instances Required](#)
- [Number of HADB Nodes Required](#)
- [HADB Storage Capacity Required](#)
- [Designing for Peak Load Compared to Steady State Load](#)

Number of Application Server Instances Required

To determine the number of applications server instances needed, evaluate your environment on the basis of the factors explained in [“Estimating Load on Application Server Instances” on page 24](#). Each application server instance can use more than one Central Processing Unit (CPU) and should have at least one CPU allocated to it.

Number of HADB Nodes Required

As a general guideline, you should plan to have one HADB node for each CPU in your system. For example, use two HADB nodes for a machine that has two CPUs.

NOTE	If you have more than one HADB node per machine (for example if you are using bigger machines), then you must ensure that there is enough redundancy and scalability on the machines such as, multiple uninterruptible power supplies, independent disk controllers.
-------------	--

HADB Storage Capacity Required

The HADB provides near-linear scaling with the addition of more nodes, until you exceed the network capacity. Each node must be configured with storage devices on a dedicated disk or disks. All nodes must have equal space allocated on the storage devices. Make sure that the storage devices are allocated on local disks.

For example, suppose the expected session data is X MB. The HADB replicates the data on mirror nodes, and therefore, 2X MB of storage is needed.

Further, the HADB uses indexes to enable fast access to data. An additional 2X MB is required (for both nodes together) for indexes (assuming a less than 100% fillings rate). This implies that a storage capacity of 4X is required.

Therefore, the expected storage capacity needed by the HADB is four times the expected data volume.

If the system has to be designed for future expansion (by adding bigger disks to nodes or adding new nodes to the system) without loss of data from the HADB, the expected storage capacity is eight times the expected data volume. This is because, for online upgrades you might want to refragment the data after adding new nodes. In this case, you will need a similar amount (4X) of additional space on the data devices, thus increasing the total storage capacity required to 8X.

Additionally, HADB uses disk space for internal use as follows:

- Space for temporary storage of log buffer. This space is four times the `logBufferSize`. The `logBufferSize` is the size of the log buffer, which keeps track of operations related to data.

NOTE The default value of `logBufferSize` is 48 MB.

- Space for internal administration purpose. This space is one percent of the storage device size.

For more information, see *Sun Java System Application Server Administration Guide* and *Sun Java System Application Server Performance Tuning Guide*.

The following table summarizes the HADB storage space requirements for a session data of X MB. The left row lists the condition (whether online addition or removal of HADB nodes is required), and the right row lists the HADB storage space required.

Table 2-3 HADB Storage Space Requirement for Session Size of X MB

Condition	HADB Storage Space Required
Addition or removal of HADB nodes while online is <i>not</i> required.	(4X MB) + (4*logBufferSize) + (1% of Device Size)
Addition or removal of HADB nodes while online is required.	(8X MB) + (4*logBufferSize) + (1% of Device Size)

NOTE If the HADB runs out of device space, error codes 4593 or 4592 are returned and error messages are written to the history files. For more information on these messages, see *Sun Java System Application Server Troubleshooting Guide*.

If the HADB runs out of device space, client requests to insert or update data are not accepted. However, delete operations are accepted.

Setting Data Device Size

Use the following command to set the size of the data devices of the HADB:

```
hadbm set TotalDatadeviceSizePerNode
```

The `hadbm` command restarts all the nodes, one by one, for the change to take effect. For more information on configuring the HADB, see *Sun Java System Application Server Administration Guide*.

NOTE	The current version of the <code>hadbm</code> command does not add data devices to a running HADB database.
-------------	---

Designing for Peak Load Compared to Steady State Load

In a typical deployment, there is a difference between steady state and peak workloads.

If you design for peak load, you must deploy a system that can sustain the expected maximum load of users and requests without a degradation in response time. This implies that your system can handle extreme cases of expected system load.

If the difference between peak load and steady state load is substantial, designing for peak loads may mean that you are spending on resources that will be idle for a significant amount of time.

If you design for steady state load, then you don't have to deploy a system with all the resources required to handle the server's expected peak load. However a system designed to support steady load will have slower response time when peak load occurs.

Frequency and Duration of Peak Load

The factor that may affect whether you want to design for peak load or for steady state is how often your system is expected to handle the peak load. If peak load occurs several times a day or even per week, you may decide that this is enough time to warrant expanding capacity to handle this load. If the system operates at steady state 90 percent of the time, and at peak only 10 percent of the time, then you may prefer to deploy a system designed around steady state load.

This implies that your system's response time will be slower only 10 percent of the time. Decide if the frequency or duration of time that the system operates at peak justifies the need to add resources to your system (should this be required to handle peak load).

Planning Network Configuration to Meet Your Performance Goals

When planning how to integrate Sun Java System Application Server into your network for optimal performance, you should estimate the bandwidth requirements and plan your network in such a way that it can meet your performance requirements.

The following topics are covered in this section:

- [Estimating Bandwidth Requirements](#)
- [Calculating Bandwidth Required](#)
- [Estimating Peak Load](#)
- [Configuring Subnets](#)
- [Choosing Network Cards](#)
- [Network Settings for HADB](#)
- [Identifying Failure Classes](#)

Estimating Bandwidth Requirements

When you decide on the desired size and bandwidth of your network, first determine your network traffic and identify its peak. Check if there is a particular hour, day of the week, or day of the month when overall volume peaks, and then determine the duration of that peak.

TIP	At all times consult network experts at your site about the size and type of network components you are considering.
------------	--

During peak load times, the number of packets in the network is at its highest level. In general, if you design for peak load, scale your system with the goal of handling 100 percent of peak volume. Bear in mind, however, that any network behaves unpredictably and that despite your scaling efforts, 100 percent of peak volume might not always be handled.

For example, assume that at peak load, five percent of your users occasionally do not have immediate Internet access when accessing applications deployed on Application Server. Of that five percent, determine how many users retry access after the first attempt. Again, not all of those users may get through, and of that unsuccessful portion, another percentage will retry. As a result, the peak appears longer because peak use is spread out over time as users continue to attempt access.

To ensure optimal access during times of peak load, start by verifying that your Internet service provider (ISP) has a backbone network connection that can reach an Internet hub without degradation.

Calculating Bandwidth Required

Based on the calculations you made in [“Establishing Performance Goals” on page 23](#), you should determine the additional bandwidth required for deploying Sun Java System Application Server at your site.

Depending on your method of access (T-1 lines, ISDN, and so on), you can calculate the amount of increased bandwidth you require to handle your estimated load. For example, suppose your site uses T-1 or higher-speed T-3 links for Internet access. Given their bandwidth, you can estimate how many lines you will need on your network, based on the average number of requests generated per second at your site and the maximum peak load. You can calculate these figures using a web site analysis- and monitoring-tool.

Example Calculation of Bandwidth Required

A single T-1 line can handle 1.544 Mbps. Therefore, a network of four T-1 lines carrying 1.544 Mbps each can handle approximately 6 Mbps of data. Assuming that the average HTML page sent back to a client is 30 kilobytes (KB), this network of four T-1 lines can handle the following traffic per second:

$6,176,000 \text{ bits} / 8 \text{ bits} = 772,000 \text{ bytes per second}$

$772,000 \text{ bytes per second} / 30 \text{ KB} = \text{approximately } 25 \text{ concurrent client requests for pages per second.}$

With a traffic of 25 pages per second, this system can handle 90,000 pages per hour (25 x 60 seconds x 60 minutes), and therefore 2,160,000 pages per day maximum, assuming an even load throughout the day. If the maximum peak load is greater than this, you will have to increase the bandwidth accordingly.

Estimating Peak Load

Having an even load throughout the day is probably not realistic. You need to determine when peak load occurs, how long it lasts, and what percentage of the total load is the peak load.

Example Calculation of Peak Load

If peak load lasts for two hours and takes up 30 percent of the total load of 2,160,000 pages, this implies that 648,000 pages must be carried over the T-1 lines during two hours of the day.

Therefore, to accommodate peak load during those two hours, you should increase the number of T-1 lines according to the following calculations:

$648,000 \text{ pages} / 120 \text{ minutes} = 5,400 \text{ pages per minute}$

$5,400 \text{ pages per minute} / 60 \text{ seconds} = 90 \text{ pages per second}$

If four lines can handle 25 pages per second, then approximately four times that many pages requires four times that many lines, in this case 16 lines. The 16 lines are meant for handling the realistic maximum of a 30 percent peak load.

Obviously, the other 70 percent of your load can be handled throughout the rest of the day by these many lines.

Configuring Subnets

If you use the separate tier topology, where the application server instances and HADB nodes are on separate tiers, you can achieve a performance improvement by keeping HADB nodes on a separate subnet. This is because HADB uses the User Datagram Protocol (UDP). Using a separate subnet reduces the UDP traffic on the machines outside of that subnet.

Choosing Network Cards

For greater bandwidth and optimal network performance, use at least 100 Mbps Ethernet cards or, preferably, 1 Gbps Ethernet cards between servers hosting Sun Java System Application Server and the HADB nodes, as well as among other resources such as HADB databases that are hosted on other machines.

Network Settings for HADB

Use the following suggestions to make HADB work optimally in the network:

- Use switched routers so that each network interface has a dedicated 100 Mbps or better Ethernet channel.
- If you are running HADB on a multi-CPU machine hosting four or more HADB nodes, use 1 Gbps Ethernet cards.

NOTE If the average session size is greater than 50 KB, use 1 Gbps Ethernet cards even if there are less than four HADB nodes per machine.

- If you suspect network bottlenecks within HADB nodes:
 - Run network monitoring software on your HADB servers to diagnose the problem.
 - Consider replacing any 100 Mbps Ethernet cards in the network with 1 Gbps Ethernet cards.
- The current release of HADB is not generally capable of running on servers with multiple network interface cards. If you need network bandwidth beyond what can be offered with a single network interface card per computer, consult Sun customer support for alternative solutions.

Planning for Availability

Availability must be planned according to the application and customer requirements.

There are two ways to achieve high availability:

- [Adding Redundancy to the System](#)

- [Using Multiple Clusters to Improve Availability](#)

Adding Redundancy to the System

One way to achieve high availability is to add redundancy to the system—redundancy of hardware and software. When one unit fails, the redundant unit takes over. This is also referred to as fault tolerance.

In general, to achieve high availability, you should determine and remove every possible point of failure in the system.

This section discusses the following topics:

- [Identifying Failure Classes](#)
- [Using Redundancy Units to Improve Availability](#)
- [Using Spare Nodes to Improve Fault Tolerance](#)
- [Planning Failover Capacity](#)

Identifying Failure Classes

The level of redundancy is determined by the failure classes (types of failure) that the system needs to tolerate. Some examples of failure classes are: system process, machine, power supply, disk, network failures, building fires and catastrophes.

Duplicated system processes tolerate single system process failures. Duplicated machines tolerate single machine failures. Attaching the duplicated mirrored (paired) machines to different power supplies tolerates single power failures. By keeping the mirrored machines in separate buildings, a single-building fire can be tolerated and by keeping them in separate geographical locations, natural catastrophes like earth quake in a location can be tolerated.

When planning availability, you should determine the failure classes covered by the system.

Using Redundancy Units to Improve Availability

To improve availability, HADB nodes are always used in Data Redundancy Units (DRUs) as explained in [“Introducing HADB” on page 19](#).

Using Spare Nodes to Improve Fault Tolerance

The use of spare nodes as explained in [“Spare Nodes” on page 21](#) improves fault tolerance. Although spare nodes are not mandatory, their use is recommended for maximum availability.

Planning Failover Capacity

Failover capacity planning implies deciding how many additional servers and processes you need to add to Sun Java System Application Server installation so that in the event of a server or process failure, the system can seamlessly recover data and continue processing. If your system gets overloaded, a process or server failure might result, causing response time degradation or even total loss of service. Preparing for such an occurrence is critical to successful deployment.

To maintain capacity, especially at peak loads, we recommended that you add spare machines running Application Server instances to your existing Application Server installation. For example, assume you have a system with two machines running one Application Server instance each. Together, these machines can handle a peak load of 300 requests per second. If one of these machines becomes unavailable, the system will be able to handle only 150 requests, assuming an even load distribution between the machines. Therefore half the requests during peak load would not be served.

Using Multiple Clusters to Improve Availability

To improve availability, instead of using a single cluster, you should group the application server instances into multiple clusters. This way, you can perform online upgrades for clusters (one by one) without loss of service.

For more information on setting up multiple clusters and using multiple clusters to perform online upgrades without loss of service, see *Sun Java System Application Server Administration Guide*.

Selecting a Topology

After estimating the factors related to performance as explained in [Chapter 2, “Planning your Environment,”](#) you should decide the *topology* that you will use to deploy Sun Java™ System Application Server 7 Enterprise Edition.

A topology is the schematic arrangement of Application Server components (machines, application server instances, and HADB nodes), and the communication flow between these components.

This chapter describes the following two recommended topologies:

- Co-located: The application server instance and the HADB node are on the same machine.
- Separate Tier: The application server instance and the HADB node are on different machines.

Both the topologies have common building blocks—multiple application server instances that form a cluster, a mirrored set of HADB nodes, and HADB spare nodes. Both of them require a set of common configuration settings to function properly.

This chapter describes the following topics:

- [Common Requirements](#)
- [Co-located Topology](#)
- [Separate Tier Topology](#)
- [Comparison of Topologies](#)

Common Requirements

The following topics in this section describe the requirements that are common to both the topologies:

- [General Requirements](#)
- [HADB Nodes and Machines](#)
- [Load Balancer Configuration](#)

General Requirements

- Machines that host HADB nodes must be provided in pairs.
- Each DRU must have the same number of machines. You must create the HADB database in such a way that the mirrored (paired) nodes are on a different DRU than the primary nodes.
- Each machine that hosts HADB nodes must have local disk storage, which is used to store all persisted information in the HADB.
- Machines that host the HADB nodes must run the same operating system. These machines should be as identical as possible in terms of configuration and performance.
- For HTTP and SFSB session information to be persisted to the HADB, the application server instances must be in a cluster and satisfy all related requirements. For more information on configuring clusters, see *Sun Java System Application Server Administration Guide*.
- The machines hosting the application server instances should be as identical as possible in terms of configuration and performance. This is because the load balancer plug-in uses a round-robin policy for load balancing, and if you have machines of different classes hosting instances, then the load will not be balanced in the most optimum way across these machines.
- Each DRU should preferably have a separate Uninterruptible Power Supply (UPS).

HADB Nodes and Machines

Each DRU contains a complete copy of your data and can continue servicing requests in a degraded (that is, non-fault-tolerant) mode if the other DRU becomes unavailable. However, if a node in one DRU and its mirror in another DRU fail at the same time, some portion of your data is lost. For this reason, it is important that you do not set up your system such o that both DRUs can be impacted by a single failure, such as a power failure or disk failure.

NOTE Each DRU must run on a completely independent, redundant system.

- To increase capacity and throughput, add nodes in pairs with one node for each DRU.
- Assume that each machine in your configuration runs N data nodes. The failure of a single machine brings down N nodes. Therefore, for each DRU you should have N spare nodes.
- You must run the same number of HADB nodes on all machines and thereby balance load as evenly as possible.

CAUTION You should not run nodes from different DRUs on the same machine. If you must run nodes from different DRUs on the same machine, ensure that the machine can handle any single point of failure (for failures related to disk, memory, CPU, power, operating system crashes, etc.).s

Load Balancer Configuration

Both the topologies comprise application server instances in a cluster. These instances persist session information to the HADB. You must configure the load balancer to include configuration information for all the application server instances in the cluster.

For more information on setting up a cluster and adding application server instances to it, see *Sun Java System Application Server Administration Guide*.

Co-located Topology

In the co-located topology, the application server instance and the HADB nodes are on the same machine (hence the name *co-located*).

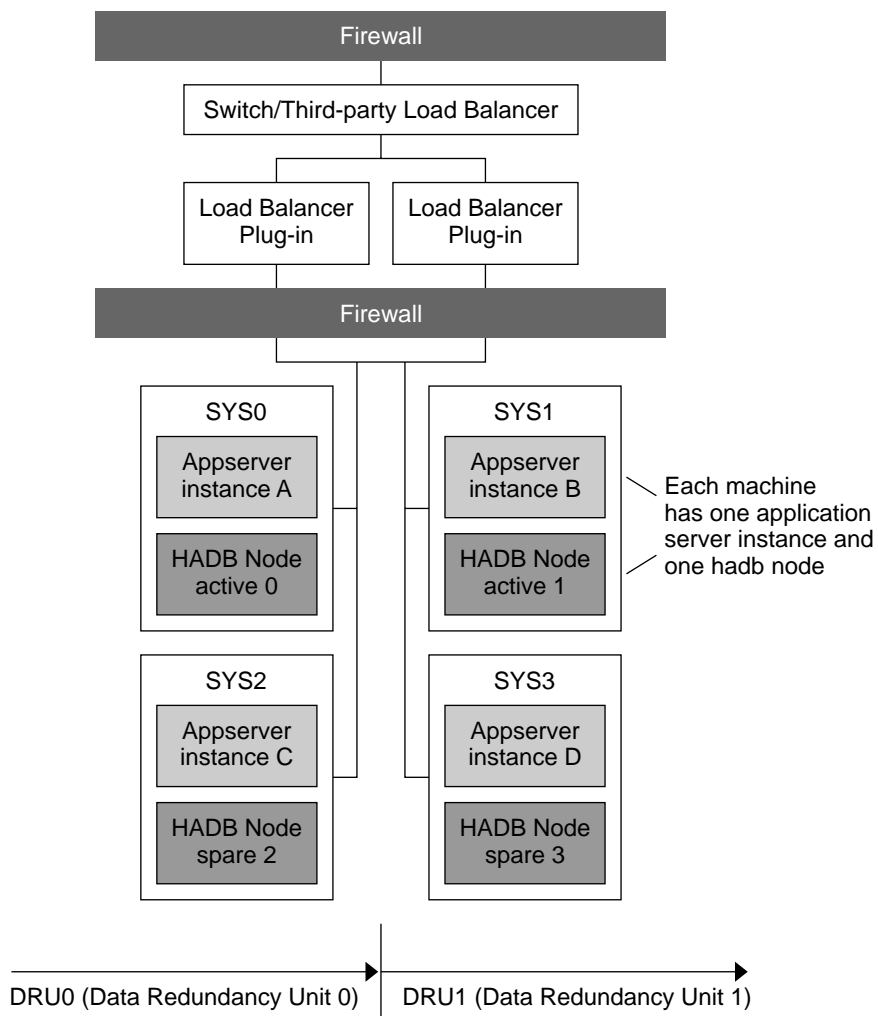
This topology requires lesser numbers of machines as compared to the separate tier topology explained later in the chapter. The co-located topology also offers improved effectiveness of CPU utilization—an application server instance and an HADB node share one machine and the processing is distributed evenly among them.

A minimum of two machines are required for this topology. To improve throughput, more machines can be added in pairs.

NOTE	The co-located topology is a good fit for large, Symmetric Multiprocessing (SMP) machines as you can take full advantage of the processing power of these machines.
-------------	---

Reference Co-located Topology

Figure 3-1 shows a reference co-located topology.

Figure 3-1 Reference Co-located Topology

Application Server instance A is on machine SYS0, Application Server instance B is on machine SYS1, Application Server instance C is on machine SYS2, and Application Server instance D is on machine SYS3.

These four instances form a cluster that persists information to the two HADB Data Redundancy Units: DRU0 and DRU1.

DRU0 comprises two machines: SYS0 and SYS2. HADB Node active 0 is on the machine SYS0. HADB Node spare 2 is on the machine SYS2.

DRU1 comprises two machines: SYS1 and SYS3. HADB Node active 1 is on the machine SYS1. HADB Node spare 3 is on the machine SYS3.

NOTE The configuration settings described in this document assume that the host names for the machines correspond to the machine names as described in the topologies. For example, for the reference co-located topology, the host names are SYS0, SYS1, SYS2, and SYS3. This applies to both the topologies (and their variations).

Configuration Settings for Reference Co-located Topology

Use the `clsetup` command for configuring the cluster (as part of the cluster configuration, the `clsetup` command creates an HADB database and sets up the JDBC connection pool and the JDBC resource for the HADB). For information on using `clsetup`, see *Sun Java System Application Server Installation Guide*.

The `clsetup` command uses the following input files:

- `clresource.conf`: This is the resource configuration file for the application server instances and the HADB.
- `clinstance.conf`: This file contains information about application server instances.

NOTE Make changes to these input files as described in the subsequent sections before you run `clsetup` command.

Changes to `clresource.conf` File

For configuration related to the topologies described in this guide, the following properties should be changed in the `clresource.conf` file:

- `hosts`: A comma separated list of host names for the machines that host HADB active nodes. For each HADB active node, include the host name of the machine. Therefore, if a machine hosts two HADB nodes, the host name of the machine must appear twice.
- `steadypoolsize`: The value of the `steadypoolsize` property is calculated using the following formula:

$$8 * (\text{number of HADB nodes}) / (\text{number of application server instances})$$

If the resulting number is a decimal, round it off to the next even number.

- **maxpoolsize:** The value of the `maxpoolsize` property is calculated using the following formula:

$$16 * (\text{number of HADB nodes}) / (\text{number of application server instances})$$

If the resulting number is a decimal, round it off to the next even number.

NOTE

- The HADB nodes include both active and spare nodes.
 - The description and the calculation of values described here apply to both the topologies (and their variations).
-

Table 3-1 describes the changes needed to the `clresource.conf` file for the reference co-located topology. The left column lists the section in the file where the property to be changed is listed, the middle column lists the property name, and the right column lists the value of the property

Table 3-1 Changes Needed to the `clresource.conf` File for Reference Co-located Topology

Section of <code>clresource.conf</code> File in Which the Property Appears	Property Name	Value
HADBINFO	hosts	SYS0,SYS1,SYS2,SYS3
JDBC_CONNECTION_POOL	steadypoolsize	8
JDBC_CONNECTION_POOL	maxpoolsize	16

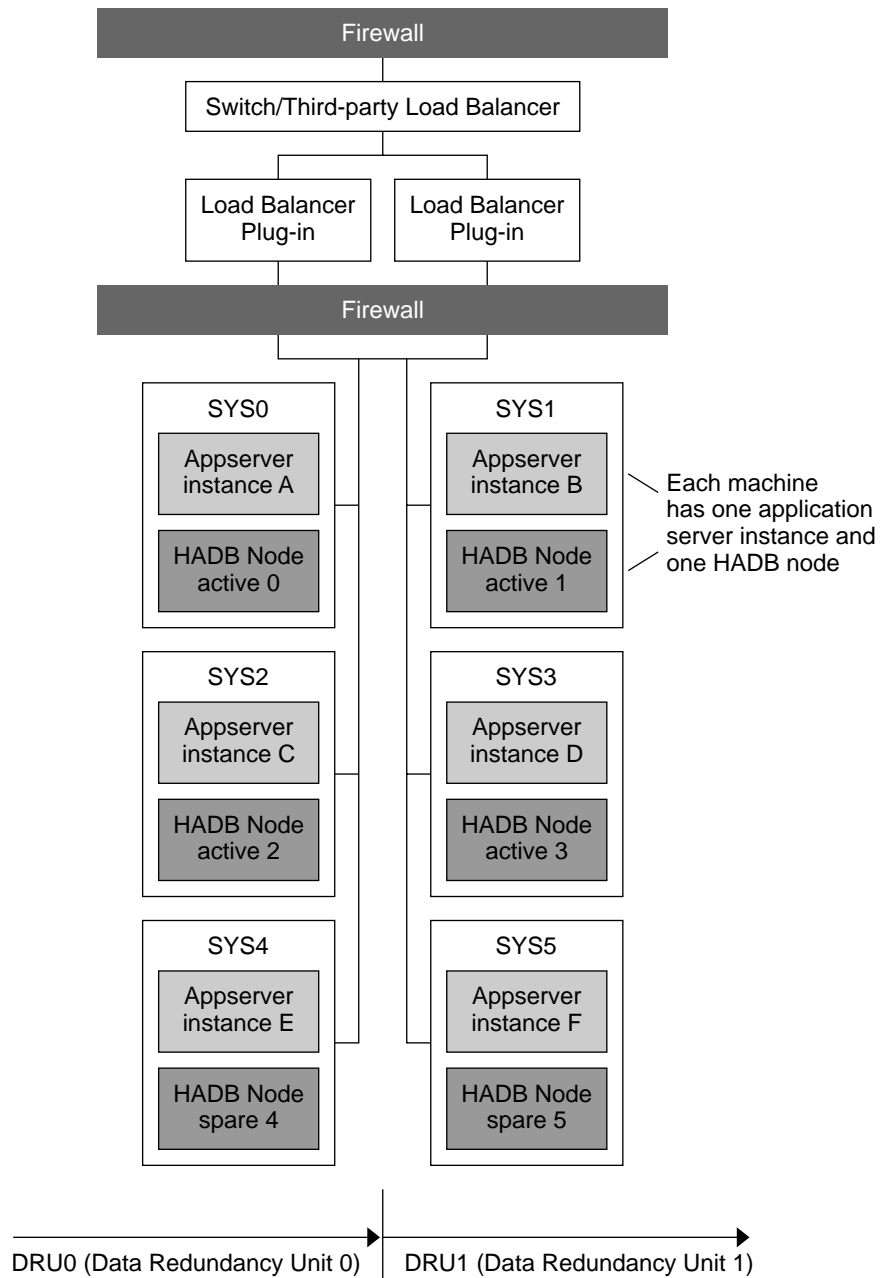
Changes to `clinstance.conf` File

In the `clinstance.conf` file, include information for each instance. For detailed information, see *Sun Java System Application Server Installation Guide*. This applies to both topologies and their variations.

Variation to Reference Co-located Topology

For better scalability and throughput, you can increase the number of application server instances and HADB nodes by adding more machines.

For example, you can add two machines, each with one application server instance and one HADB node. Make sure that you add the HADB nodes in pairs, assigning one node for each DRU. This configuration is shown in Figure 3-2.

Figure 3-2 Variation to Reference Co-located Topology

In this variation, the machines SYS4 and SYS5 have been added to the reference co-located topology described in [“Reference Co-located Topology” on page 44](#).

Application Server instance A is hosted on machine SYS0, Application Server instance B is hosted on machine SYS1, Application Server instance C is hosted on machine SYS2, Application Server instance D is hosted on machine SYS3, Application Server instance E is hosted on machine SYS4, and Application Server instance F is hosted on machine SYS5.

These instances form a cluster that persists information to the HADB Data Redundancy Units DRU0 and DRU1.

Data Redundancy Unit DRU0 comprises machines SYS0, SYS2, and SYS4. HADB Node active 0 is on the machine SYS0. HADB Node active 2 is on the machine SYS2. HADB Node spare 4 is on the machine SYS4.

Data Redundancy Unit DRU1 comprises the machines SYS1, SYS3, and SYS5. HADB Node active 1 is on the machine SYS1. HADB Node active 3 is on the machine SYS3. HADB Node spare 5 is on the machine SYS5.

Configuration Settings for Variation to the Reference Co-located Topology

Make the changes as described in the subsequent sections before you run the `clsetup` command.

Changes to clresource.conf File

Table 3-2 describes the changes needed to the `clresource.conf` file for the variation to the reference co-located topology as described in this section. The left column lists the section in the file in which the property to be changed is listed, the middle column lists the property name, and the right column lists the value of the property. For more information on the values of these properties, see [“Changes to clresource.conf File” on page 46](#).

Table 3-2 Changes Needed to `clresource.conf` File for Variation to Reference Co-located Topology

Section of <code>clresource.conf</code> File in Which the Property Appears	Property Name	Value
HADBINFO	hosts	SYS0,SYS1,SYS2,SYS3,SYS4,SYS5
JDBC_CONNECTION_POOL	steadypoolsize	8
JDBC_CONNECTION_POOL	maxpoolsize	16

Changes to clinstance.conf File

In the `clinstance.conf` file, include the information for each instance. For more information, see *Sun Java System Application Server Installation Guide*.

Separate Tier Topology

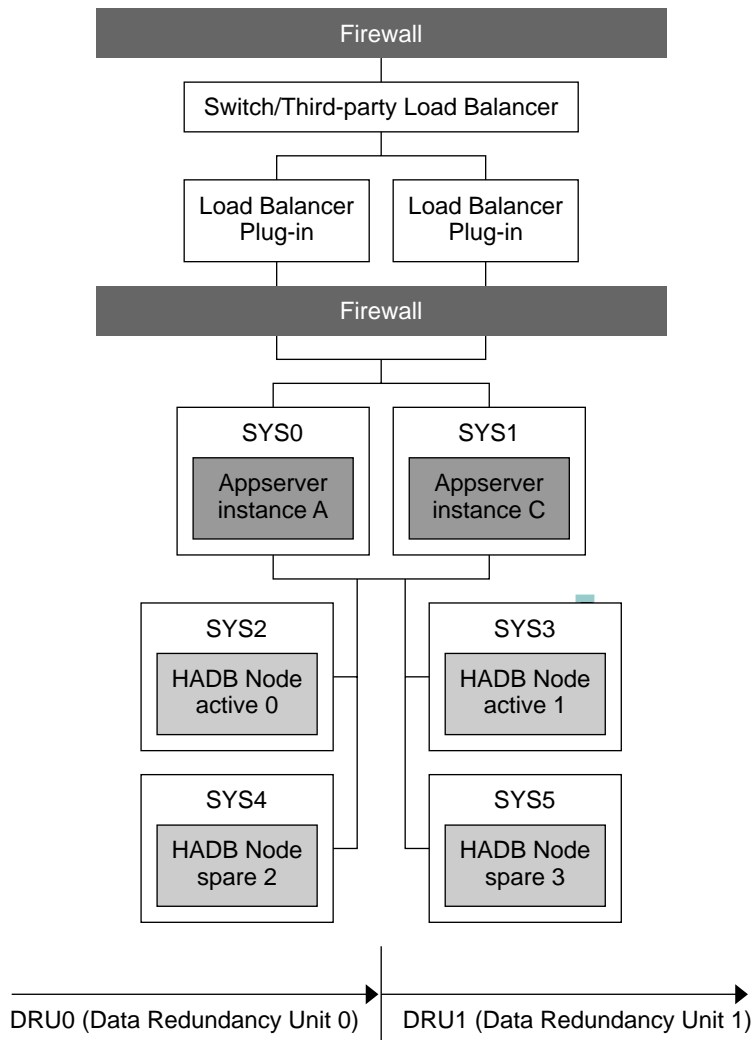
In this topology, Application Server instances and the HADB nodes are on different machines (hence the name *separate tier*).

This topology requires more hardware than the co-located topology. This topology may be a good fit if you have different types of machine—you can allocate a different set of machines to the Application Server instance tier and to the HADB nodes tier. For example, you can use more powerful machines for the Application Server instances tier and less powerful machines for the HADB tier.

Reference Configuration

Figure 3-3 shows the reference separate tier topology.

Figure 3-3 Reference Separate Tier Topology



In this reference topology, the Application Server instance A is hosted on machine SYS0 and the Application Server instance B is hosted on the machine SYS1.

These two instances form a cluster that persists session information to Data Redundancy Units DRU0 and DRU1.

The Data Redundancy Unit DRU0 comprises two machines: SYS2 and SYS4. The HADB Node active 0 is on machine SYS2 and the HADB Node spare 2 is on machine SYS4.

The Data Redundancy Unit DRU1 comprises two machines SYS3 and SYS5. The HADB Node active 1 is on machine SYS3 and the HADB Node spare 3 on machine SYS5.

All the nodes on a DRU are on different machines, so that even if one machine becomes unavailable, the complete data for any DRU continues to be available on other machines.

Configuration Settings for Reference Separate Tier Topology

Make the changes as described in the subsequent sections to these input files before you run the `clsetup` command.

Changes to `clresource.conf` File

Table 3-3 describes the changes needed to the `clresource.conf` file for the reference separate tier topology. The left column lists the section in the file in which the property to be changed is listed, the middle column lists the property name, and the right column lists the value of the property. For more information on the values of these properties, see [“Changes to `clresource.conf` File” on page 46](#).

Table 3-3 Changes Needed to `clresource.conf` File for Reference Separate Tier Topology

Section of <code>clresource.conf</code> File in Which the Property Appears	Property Name	Value
HADBINFO	hosts	SYS2,SYS3,SYS4,SYS5
JDBC_CONNECTION_POOL	steadypoolsize	16
JDBC_CONNECTION_POOL	maxpoolsize	32

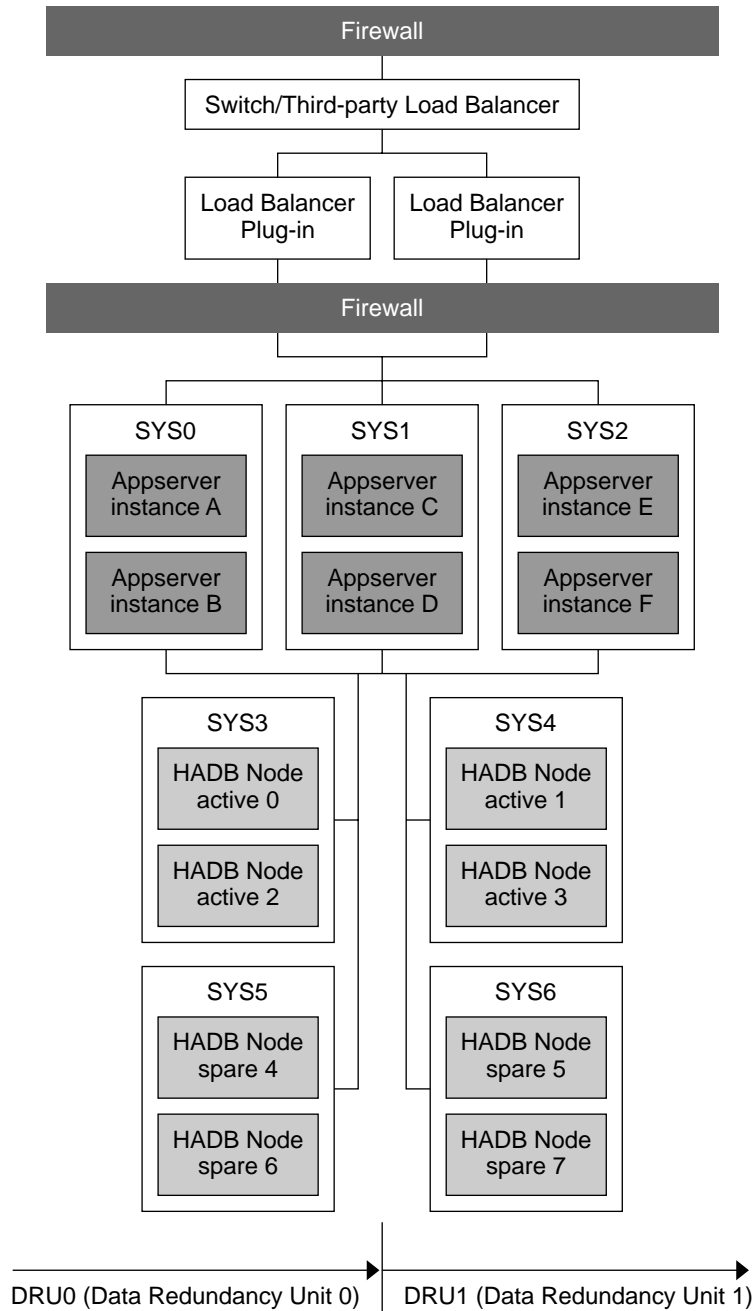
Changes to `clinstance.conf` File

In the `clinstance.conf` file, include information for each instance. For more information, see *Sun Java System Application Server Installation Guide*.

Variation to Reference Separate Tier Topology

You can increase the number of Application Server instances by adding more machines horizontally to the configuration. For example, you can add another machine to the reference configuration by creating a new Application Server instance. Similarly, you can increase the number of HADB nodes by adding more machines to host HADB nodes. Make sure that you add the HADB nodes in pairs with one node for each DRU.

This configuration is shown in Figure 3-4.

Figure 3-4 Variation to Reference Separate Tier Topology

In this configuration, each machine hosting Application Server instances has two Application Server instances. There are thus a total of six Application Server instances in the cluster.

The HADB nodes are on machines SYS3, SYS4, SYS5, and SYS6.

The Data Redundancy Unit DRU0 comprises two machines: SYS3 and SYS5. HADB Node active 0 and the HADB Node active 2 are on machine SYS3. HADB Node spare 4 and the HADB Node spare 6 are on machine SYS5.

The Data Redundancy Unit DRU1 comprises two machines SYS4 and SYS6. HADB Node active 1 and HADB Node active 3 are on machine SYS4. HADB Node spare 5 and HADB Node spare 7 are on machine SYS6.

Each machine hosting HADB nodes hosts two HADB nodes each. There are thus a total of eight HADB nodes (four active nodes and four spare nodes).

Configuration Settings for Variation to Reference Separate Tier Topology

Make the changes as described in the subsequent sections before you run the `clsetup` command.

Changes to cresource.conf File

Table 3-4 describes the changes needed to the `cresource.conf` file for the variation to the reference separate tier topology. The left column lists the section in the file in which the property to be changed is listed, the middle column lists the property name, and the right column lists the value of the property. For more information on the values of these properties, see [“Changes to cresource.conf File” on page 46](#).

Table 3-4 Changes Needed to cresource.conf File for Variation to Reference Separate Tier Topology

Section of cresource.conf File in Which the Property Appears	Property Name	Value
HADBINFO	hosts	SYS3,SYS4,SYS3,SYS4, SYS5,SYS6,SYS5,SYS6
JDBC_CONNECTION_POOL	steadypoolsize	12
JDBC_CONNECTION_POOL	maxpoolsize	22

Changes to clinstance.conf File

In the `clinstance.conf` file, include information for each Application Server instance. For more information, see *Sun Java System Application Server Installation Guide*.

Comparison of Topologies

Table 3-5 presents a comparison of the co-located topology and the separate tier topology. The left column lists the name of the topology, the middle column lists the advantages of the topology, and the right column lists the disadvantages of the topology

Table 3-5 Comparison of Topologies

Topology	Advantages	Disadvantages
Co-located Topology	<ul style="list-style-type: none"> Requires fewer numbers of machines as compared to the separate tier topology. Because the HADB nodes and the application server instances are on the same tier, you can create an application server instance on each spare node to handle additional load. Improved effectiveness of CPU utilization. An application server instance and an HADB node share one machine and the processing is distributed evenly among them. Useful for large, Symmetric Multiprocessing (SMP) machines as you can take full advantage of the processing power of these machines. 	Increased complexity of maintenance. For example, if you want to perform maintenance tasks (that require shutting down of machines) on HADB nodes, the application server instances on the machine hosting HADB nodes also become unavailable while the machine is unavailable.

Table 3-5 Comparison of Topologies

Topology	Advantages	Disadvantages
Separate Tier Topology	<ul style="list-style-type: none">• Easier maintenance. For example, you can perform maintenance tasks for the machines that host application server instances without having to bring down HADB nodes.• Useful in situations where you have different types of machines. You can allocate a different set of machines to the application server instances tier and to the HADB tier. For example, you can use the more powerful machines for the application server instances tier and the less powerful machines for the HADB tier.	<ul style="list-style-type: none">• Requires more machines as compared to the co-located topology. Because application server instances and HADB nodes are located on separate tiers, application server instances cannot be located on the machines that host the HADB spare nodes.• Reduced effectiveness of CPU utilization. The tier consisting of application server instances and the tier consisting of HADB nodes will likely have uneven loads. This is more significant when the number of machines is smaller (four to six).

Determining Which Topology to Use

You should test the different topologies mentioned in this chapter and experiment with different combinations of machines and CPUs to determine which topology (or its variation) best meets your performance and availability requirements.

Determine what trade offs you want to make to serve your needs the best. For example, if ease of maintenance is a critical requirement for you, the separate tier topology is more suitable. However, you will have to use a higher number of machines as compared to the co-located topology.

An important factor in the choice of topology is the type of machines you have in your setup. If you have large, Symmetric Multiprocessing (SMP) machines in your system, the co-located topology is an attractive option because you can take full advantage of the processing power of these machines. If you have different types of machines, separate tier topology may be more useful because you can allocate a different set of machines to the application server instances tier and to the HADB tier. For example, you can use more powerful machines for the application server instances tier and the less powerful machines for the HADB tier.

Checklist for Deployment

This appendix provides a checklist to get started on the evaluation and production with Sun Java System Application Server 7 2004Q2.

Table 0-1 Checklist

Component/Feature	Description
Application	<p>Determine the following requirements for the application to be deployed.</p> <ol style="list-style-type: none"> 1. The required/acceptable response time. 2. Peak load characteristics. 3. Necessary persistence scope and frequency. 4. Session timeout in web.xml. 5. The failover and availability requirements. <p>For more information on planning for your requirements, see Chapter 2, "Planning your Environment" and "About Application Server Performance" in <i>Sun Java System Application Server 7 Performance and Tuning Guide</i>.</p>
Hardware	<p>Determine the following hardware requirements for deploying the application:</p> <ol style="list-style-type: none"> 1. The same type of hardware should be used to host HADB nodes. 2. Have necessary amounts of hard disk space and memory installed. 3. Use the sizing exercise to identify the requirements for your deployment. <p>For detailed information on the minimum system requirements, see <i>Sun Java System Application Server 7 Release Notes</i> at the product documentation web site at http://docs.sun.com/db/prod/s1appsrv#hic</p>

Table 0-1 Checklist

Component/Feature	Description
Operating System	<ol style="list-style-type: none"> 1. Ensure that the product is being installed on a supported platform. 2. Ensure that the patch levels are up-to-date and accurate. <p>For more information on the supported platforms, see <i>Sun Java System Application Server 7 Release Notes</i> at the product documentation web site at http://docs.sun.com/db/prod/s1appsrv#hic</p>
Network Infrastructure	<ol style="list-style-type: none"> 1. Identify single points of failures and address them. 2. Make sure that the NIC cards and other network components are correctly configured. 3. Run <code>ttcp</code> benchmark test to determine if the throughput meets the requirements/expected result. 4. Setup <code>rsh/ssh</code> based on your preference so that HADB nodes can be installed. <p>For more information on the required network infrastructure, setting up <code>rsh/ssh</code>, see <i>Sun Java System Application Server 7 Installation Guide</i>.</p>
Back-ends and other external datasources	<p>Check with your domain expert/vendor to ensure that these datasources are configured appropriately.</p>
System Changes/Configuration	<ol style="list-style-type: none"> 1. Make sure that changes to <code>/etc/system</code> and its equivalent on Linux are completed before running any performance/stress tests. 2. Make sure you have completed changes to TCP/IP settings. 3. By default, the system comes with lots of services pre-configured. Not all of them are required to be running. Turning off services that you do not need and thereby conserve system resources. 4. On Solaris, use <code>Setoolkit</code> to determine the behavior of the system. Resolve any flags that show up. <p>For more information on optimum system configuration, see "About Application Server Performance" in <i>Sun Java System Application Server 7 Performance and Tuning Guide</i>.</p>

Table 0-1 Checklist

Component/Feature	Description
Application Server and HADB Installation	<ol style="list-style-type: none">1. Ensure that these servers are not installed on NFS mounted volumes.2. Check for enough disk space and RAM when installing both Application Server and the HADB nodes on the same machine.3. Check for enough independent disks when installing multiple HADB nodes on the same system. <p>For more information on installing the Application Server and the HADB, see <i>Sun Java System Application Server 7 Installation Guide</i>.</p>
HADB Configuration	<ol style="list-style-type: none">1. Set the size of the HADB Data Device.2. Define the DataBufferPoolSize.3. Define the LogBufferSize.4. Define the InternalBufferSize.5. Set the NumberOfLocks.6. Set optimum time-out values for various Application Server components.7. Create the Physical layout of HADB nodes on the filesystem. <p>For more information on installing the HADB, see chapter on “Preparing for HADB Setup” in <i>Sun Java System Application Server Installation Guide</i>.</p> <p>For more information on tuning the HADB, see chapter on “Tuning for High-Availability” in <i>Sun Java System Application Server Performance and Tuning Guide</i>.</p>

Table 0-1 Checklist

Component/Feature	Description
Application Server Configuration	<ol style="list-style-type: none">1. If sso is not required, then it can be turned off via <code><sso-enabled = false /></code> property in <code>server.xml</code> configuration file.2. Similarly, if MDB are not being used, then <code>jms-service</code> can be turned off.3. Logging: Enable access log rotation.4. Choose the right logging level, WARNING would be appropriate most of the times.5. Configure J2EE containers using Admin Console.6. Configure HTTP listeners using Admin Console.7. Configure ORB threadpool using Admin Console.8. If the application to be deployed has EJB's, then ensure that the <code>UtilDelegate</code> flag is used. For information on using this flag, see "Enabling the High Performance CORBA Util Delegate," in <i>Sun Java System Application Server Performance and Tuning Guide</i>.9. If using Type2 Drivers or calls involving native code, ensure that <code>mtmalloc.so</code> is specified in the <code>LD_LIBRARY_PATH</code>.10. Consider disabling <code>server.policy</code> file if performance is critical.11. Ensure that the appropriate persistence scope and frequency are used and they are not overridden underneath in the individual Web/EJB modules.12. Ensure that only critical methods in the SFSB are checkpointed. <p>For more information on tuning Application Server settings, see <i>Sun Java System Application Server 7 Performance and Tuning Guide</i>.</p> <p>For more information on configuring various Application Server components and services, see <i>Sun Java System Application Server 7 Administration Guide</i>.</p>

Table 0-1 Checklist

Component/Feature	Description
Load balancer Configuration	<ol style="list-style-type: none">1. Make sure you have installed the Web Server.2. Make sure you have installed the load balancer plug-in. For more information on installing load balancer plug-in, see chapter, "Installing Standard and Enterprise Edition Software" in <i>Sun Java System Application Server 7 Installation Guide</i>.3. Ensure to turn off unnecessary functions.4. Make sure you have disabled patch checks.5. Make sure you have configured <code>RqThrottle</code>, <code>KeepAliveQuery</code>* parameters. Lower the <code>KeepAliveQuery</code>* values; lower the value, latency will be lower on lightly loaded systems. Higher the values, higher will be the throughput on highly loaded systems.6. Make sure you have enabled and configured <code>perfdump</code> in <code>instancename-obj.conf</code> file. <p>For more information on load balancer configuration, see chapter "Configuring HTTP Load Balancing and Failover" in <i>Sun Java System Application Server 7 Administration Guide</i>.</p>
JVM Configuration	<ol style="list-style-type: none">1. At a minimum, specify the minimum and maximum heap sizes to be the same, and equal to one GB for each instance.2. Refer to documentation on http://java.sun.com for configurable options in Java Hotspot JVM.3. When running multiple instances of Application Server, consider creating a processor set and bind the Application Server to it. This helps in cases where the CMS collector is used to sweep the old generation.

Table 0-1 Checklist

Component/Feature	Description
Configuring time-outs in Load balancer	<ol style="list-style-type: none"> 1. Response-time-out-in-seconds: Determines how much time the load balancer will wait before declaring an Application Server instance as unhealthy. This value needs to be set based on the response time characteristics of your application. If this value is too high, then the Web Server/Load balancer plug-in is going to wait for a long time before marking that Application Server instance as unhealthy. If the value for Response-time-out-in-seconds is set too low and if the Application Server's response time crosses this threshold, the instance will be incorrectly marked as unhealthy. 2. Interval-in-seconds: Specifies the time interval in seconds after which unhealthy instances will be checked to find out if they have returned to a healthy state. Too low a value will generate extra traffic from the load balancer plug-in to Application Server instances and too high a value will delay the routing of requests to the instance that has turned healthy. 3. Timeout-in-seconds: Specifies the duration for a response to be obtained for a health check request. This value needs to be adjusted based on the traffic among the systems in the cluster to ensure that the health check succeeds. <p>For more information on these load balancer configuration parameters, see chapter, "Configuring HTTP Load Balancing and Failover" in <i>Sun Java System Application Server 7 Administration Guide</i>.</p>
Configuring time-outs in Application Server	<ol style="list-style-type: none"> 1. Max-wait-time-millis: Defines the wait time to get a connection from the pool before throwing an exception. Default is 60000 milli seconds. Consider changing this value in case of highly loaded systems where the size of the data being persisted is greater than 50 Kb. 2. Cache-idle-timeout-in-seconds: Applies to entity beans and stateful session beans. Defines the time the bean is allowed to be idle in the cache before it gets passivated. 3. Removal-timeout-in-seconds: The amount of time that the bean remains passivated, that is, idle in the backup store, is controlled by removal-timeout-in-seconds parameter. The default value is 60 minutes. This value needs to be adjusted based on the need for SFSB failover. 4. All of these values should be set by paying attention to the HADB's JDBC connection pool setting max-wait-time-in-millis. <p>For more information on tuning the timeout parameters, see chapter, "Tuning the Application Server" in <i>Sun Java System Application Server 7 Performance and Tuning Guide</i>.</p>

Table 0-1 Checklist

Component/Feature	Description
HADB time-outs	<ol style="list-style-type: none"> <li data-bbox="672 267 1348 708">1. <code>sql_client_timeout</code>: This variable controls the wait time of SQLSUB for an idle client. For example, a client which has logged on, sends some requests, and then waits for user input. A client that has been idle for more than 30 minutes is assumed to be dead, and the session is terminated. Setting the value too low may cause SQL sessions to be aborted prematurely, while setting it too high may cause SQL sessions that are not idle, but has exited, to occupy resources. This in turn may prevent other SQL clients from logging on. When tuning this variable also consider the settings of “<code>nsessions</code>.” The default value is 1800 seconds and it can be changed by editing the configuration file. If the HADB JDBC connection pool <code>steady-pool-size</code> is greater than(>) <code>max-pool-size</code>, then <code>idle-timeout-in-seconds</code> should be set lower than the <code>sql_client_timeout</code>, so that the Application Server itself will close the connection before HADB closes the connection. <li data-bbox="672 718 1348 1025">2. <code>lock_timeout</code>: Specifies the maximum time a transaction waits for access to data. When this time is exceeded, the transaction generates the error message: “The transaction timed out.” Such time-outs are caused by transactions waiting for locks held by other transactions (i.e. deadlocks), and causing high server load. The default value is 5000 ms. Do not set this value to below 500 ms. If you see the “transaction timed out” messages in the server log, then it is a good idea to increase this value. The lock timeout value can be set by adding a property to the <code>ha jdbc</code> connection pool as: <code><property name=lockTimeout value=“x” /></code> where x is in milliseconds. <li data-bbox="672 1036 1348 1229">3. <code>Querytimeout</code>: This is the maximum time in milliseconds that the HADB waits for a query to execute. The default value is 30 seconds. If you see exceptions in the server log consistently indicating the query time out, you should consider increasing this value. This value can be set by adding a property to the <code>ha jdbc</code> connection pool as: <code><property name=QueryTimeout value=“x” /></code> where x is in milliseconds. <li data-bbox="672 1239 1348 1355">4. <code>loginTimeout</code>: This is the maximum time that the client waits to login to the HADB. This can be set by adding a property to the <code>ha jdbc</code> connection pool as: <code><property name=loginTimeout value=“x” /></code> where x is in seconds. Default value is 10sec. <li data-bbox="672 1366 1348 1529">5. <code>MaxTransIdle</code>: The maximum time a transaction can be idle (msec) between sending a reply to the client and receiving the next request. The default value is 40sec. This can be changed by adding a property to the <code>ha jdbc</code> connection pool as: <code><property name=maxtransIdle value=“x” /></code> where x is in milliseconds. <p data-bbox="672 1539 1348 1621">For more information on these timeout parameters, see chapter, “Tuning for High-Availability” in <i>Sun Java System Application Server 7 Performance and Tuning Guide</i>.</p>

Table 0-1 Checklist

Component/Feature	Description
Implications of GC in Application Server, Enterprise Edition	<p>GC pauses if they are long enough, that is greater than or equal to (\geq) 4sec, which can cause intermittent problems in persisting session state into HADB. To avoid this problem, It is recommended to tune the vm heap. In cases where even a single failure to persist data is not acceptable and also in cases where the system is not fully loaded, using CMS collector can help. Other option is to use the throughput collector.</p> <p>These can be enabled by adding:</p> <pre><jvm-options>-XX:+UseConcMarkSweepGC</jvm-options></pre> <pre><jvm-options> -XX:SoftRefLRUPolicyMSPerMB=1</jvm-options></pre> <p>Note that with the use of this option, you might experience a drop in throughput.</p> <p>For any late-breaking updates to the GC parameters, see <i>Sun Java System Application Server 7 Release Notes</i> at the product documentation web site at http://docs.sun.com/db/prod/s1appsrv#hic</p>
Common documents that can help	<p>1. JVM options: http://java.sun.com/docs/hotspot/gc1.4.2/index.html</p> <p>The following documents are available at http://docs.sun.com/db/prod/s1appsrv#hic</p> <ol style="list-style-type: none"> 1. <i>Sun Java System Application Server 7 Installation Guide</i> 2. <i>Sun Java System Application Server 7 System Deployment Guide</i> 3. <i>Sun Java System Application Server 7 System Performance and Tuning Guide</i> 4. <i>Sun Java System Application Server 7 Error Messages Guide</i> 5. <i>Sun Java System Application Server 7 Release Notes</i>

Index

A

availability 14
 for Data Redundancy Unit 43
 improving with multiple clusters 40

B

bandwidth requirements, estimating 35
 building blocks, of topology 41

C

capacity, using spare machines to maintain 40
 clinstance.conf file 47
 changes required 47
 clresource.conf file 46
 changes for reference co-located topology 46
 changes for reference separate tier topology 53
 changes for variation to co-located topology 50
 changes for variation to reference topology 56
 clsetup command 46
 clusters
 using multiple clusters to improve availability 40
 co-located topology 15, 41, 44

configuration settings for reference topology 46
 configuration settings for variation 50
 reference topology 44
 using Symmetric Multiprocessing machines 44
 variation 47

common topology requirements 42

comparison of topologies 57

configuration

 load balancer 43

configuration settings

 for reference co-located topology 46

 for reference separate tier topology 53

 for variation to co-located topology 50

 for variation to separate tier topology 56

D

Data Redundancy Unit

 ensuring availability 43

 improving availability with 39

 number of machines in 42

 power supply for 42

deployment

 about 13

 important goals 13

deployment phases

 planning your environment 15, 19

- running tests [16](#)
- selecting a topology [15](#)
- document, organization [8](#)

E

- environment planning [15](#), [19](#)
- ethernet cards [38](#)

F

- failover capacity, planning [40](#)
- failure
 - classes [39](#)
 - types [39](#)
- fault tolerance [39](#)

G

- goals, of deployment [13](#)

H

- HADB [13](#)
 - network bottlenecks [38](#)
 - network settings for [38](#)
 - nodes [43](#)
 - spare nodes [40](#)
- HADBINFO [47](#), [53](#), [56](#)
- high availability, achieving [39](#)
- host names, specifying for topology [46](#)
- hosts [47](#), [53](#), [56](#)
- hosts, value of property [46](#)

I

- intended audience [5](#)

J

- JDBC [47](#)
- JDBC_CONNECTION_POOL [47](#), [53](#), [56](#)

L

- load balancer configuration [43](#)
- local disk storage [42](#)

M

- machines
 - in Data Redundancy Unit [42](#)
 - maintaining capacity with spare machines [40](#)
- maximizing performance
 - availability [14](#)
 - response time [14](#)
 - throughput [14](#)
- maxpoolsize [47](#), [53](#), [56](#)
 - calculating [47](#)
- mirrored machines [39](#)
- multiple clusters, improving availability with [40](#)
- multiple network cards [38](#)

N

- network cards [38](#)
 - multiple [38](#)
- nodes [43](#)
 - spare [40](#)

P

peak load [35, 37](#)
 peak load times [36](#)
 planning your environment [19](#)
 pre-requisites, for using this guide [5](#)

R

recommended topologies [41](#)
 redundancy [39](#)
 response time [14](#)
 routers [37](#)
 rpm [11](#)

S

selecting, topology [15](#)
 separate tier topology [15, 41, 51](#)
 configuration settings for [53](#)
 configuration settings for variation [56](#)
 reference configuration [51](#)
 variation [54](#)
 showrev [11](#)
 spare machines, maintaining capacity with [40](#)
 spare nodes [40](#)
 improving fault tolerance with [40](#)
 steadypoolsize [47, 53, 56](#)
 calculating [46, 47](#)
 subnets [37](#)
 Sun customer support [11](#)
 Symmetric Multiprocessing machines, for co-located topology [44](#)

T

tests, running tests [16](#)
 throughput [14](#)

topology
 building blocks of [41](#)
 co-located [15, 41, 44](#)
 common requirements [42](#)
 comparison [57](#)
 determining which to use [58](#)
 recommended topologies [41](#)
 selecting [15, 41](#)
 separate tier [15, 41, 51](#)
 specifying host names for [46](#)
 types, of failure [39](#)

U

User Datagram Protocol (UDP) traffic [37](#)

