Deployment Guide

Netscape Application Server

Version 4.0

Recycled and Recyclable Paper

Version 4.0

Part Number 151-07590-00

Copyright 1999 Netscape Communications Corporation. All rights reserved

Printed in the United States of America. 00 99 98 5 4 3 2 1

Netscape Communications Corporation, 501 East Middlefield Road, Mountain View, CA 94043

Contents

Preface

This preface contains the following topics:

- Using the Documentation
- About this Guide
- How This Guide Is Organized
- Documentation Conventions

Using the Documentation

The following table lists the tasks and concepts that are described in the Netscape Application Server (NAS) and Netscape Application Builder (NAB) printed manuals and online readme file. If you are trying to accomplish a specific task or learn more about a specific concept, refer to the appropriate manual.

Note that the printed manuals are also available as online files in PDF and HTML format.

For information about	See the following	Shipped with
Late-breaking information about the software and the documentation	readme.htm	NAS 4.0 Developer Edition (Solaris), NAS 4.0, NAB 4.0
Installing Netscape Application Server and its various components (Web Connector plug-in, Netscape Application Server Administrator), and configuring the sample applications	Installation Guide	NAS 4.0 Developer Edition (Solaris), NAS 4.0

For information about	See the following	Shipped with
Installing Netscape Application Builder	install.htm	NAB 4.0
Basic features of NAS, such as its software components, general capabilities, and system architecture	Overview	NAS 4.0 Developer Edition (Solaris), NAS 4.0, NAB 4.0
Deploying Netscape Application Server at your site, by performing the following tasks: • Planning your Netscape Application Server environment • Integrating the product within your existing enterprise and network topology • Developing server capacity and performance goals • Running stress tests to measure server performance • Fine-tuning the server to improve performance	Deployment Guide	NAS 4.0

For information about	See the following	Shipped with
Administering one or more application servers using the Netscape Application Server Administrator tool to perform the following tasks: • Deploying applications with the Deployment Manager tool • Monitoring and logging server activity • Setting up users and groups • Administering database connectivity • Administering transactions • Load balancing servers • Managing distributed data synchronization	Administration Guide	NAS 4.0
Migrating your applications to the new Netscape Application Server 4.0 programming model from version 2.1, including a sample migration of an Online Bank application provided with Netscape Application Server	Migration Guide	NAS 4.0 Developer Edition (Solaris), NAS 4.0, NAB 4.0

For information about	See the following	Shipped with
Creating NAS 4.0 applications within an integrated development environment by performing the following tasks: • Creating and managing projects • Using wizards • Creating data-access logic • Creating presentation logic and layout • Creating business logic • Compiling, testing, and debugging applications • Deploying and downloading applications • Working with source control • Using third-party tools	User's Guide	NAB 4.0
Creating NAS 4.0 applications that follow the new open Java standards model (Servlets, EJBs, JSPs, and JDBC), by performing the following tasks: • Creating the presentation and execution layers of an application • Placing discrete pieces of business logic and entities into Enterprise Java Bean (EJB) components • Using JDBC to communicate with databases • Using iterative testing, debugging, and application fine-tuning procedures to generate applications that execute correctly and quickly	Programmer's Guide (Java)	NAS 4.0 Developer Edition (Solaris), NAB 4.0

For information about	See the following	Shipped with
Using the public classes and interfaces, and their methods in the Netscape Application Server class library to write Java applications	Server Foundation Class Reference (Java)	NAS 4.0 Developer Edition (Solaris), NAB 4.0
Creating NAS C++ applications using the NAS class library by performing the following tasks: • Designing applications • Writing AppLogics • Creating HTML templates • Creating queries • Running and debugging applications	Programmer's Guide (C++)	Order separately
Using the public classes and interfaces, and their methods in the Netscape Application Server class library to write C++ applications	Server Foundation Class Reference (C++)	Order separately

About this Guide

The *Deployment Guide* provides a foundation for evaluating your system needs and enterprise to ensure that you deploy Netscape Application Server (NAS) in a manner that best suits your site. Along the way, you are introduced to the general issues and concerns that you should be aware of when deploying an application server. The information is intended for system architects, technology officers, administrators, and developers who want a broader understanding of how to maximize performance of NAS and integrate it within their overall enterprise.

By purchasing NAS, your organization has just made an important technology investment. You want to be sure to leverage what NAS has to offer in light of your system and application processing goals.

How This Guide Is Organized

This guide is divided into five chapters. Read the chapters in the order they are presented, as each chapter builds on the previous one. A glossary of NAS deployment terminology is also included.

- Chapter 1, "Overview of Netscape Application Server Deployment," introduces you to deploying NAS at your site.
- Chapter 2, "Planning Your Environment,"explains how to fit NAS into your overall enterprise by establishing performance goals, assessing your overall network configuration, and understanding your options for configuring NAS topologies.
- Chapter 3, "Determining System Capacity," explains in detail how to measure NAS performance by determining system capacity, a critical step to a successful deployment. In this phase of deployment, you gather information about the level of activity that will occur on your server.
- Chapter 4, "Performance Testing and Fine-Tuning Your System," explains how to run performance tests against NAS, how to analyze the test results, and then based on the results, how to fine-tune your system to improve performance.
- Chapter 5, "Deploying the Production System," provides guidelines for bringing NAS into production.
- The "Glossary" provides definitions of terms introduced throughout this guide.

Documentation Conventions

File and directory paths are given in Windows format (with backslashes separating directory names). For Unix versions, the directory paths are the same, except slashes are used instead of backslashes to separate directories.

This guide uses URLs of the form:

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

In these URLs, *server* is the name of server on which you run your application; *domain* is your Internet domain name; *path* is the directory structure on the server; and *file* is an individual filename. Italic items in URLs are placeholders.

This guide uses the following font conventions:

- The `monospace` font is used for sample code and code listings, API and language elements (such as function names and class names), file names, path names, directory names, and HTML tags.
- *Italic* type is used for book titles, emphasis, variables and placeholders, and words used in the literal sense.

Chapter 1

Overview of Netscape Application Server Deployment

This chapter introduces you to deploying Netscape Application Server (NAS), an application server that enables enterprises to easily develop, implement, and manage business-critical Internet applications. As a NAS customer, you are assured of a solid development, deployment, and administrative platform with comprehensive legacy and enterprise integration solutions.

This chapter describes briefly what you need to know to set up your NAS system in the best way possible—that is, the way that best meets your application, transaction, and security needs.

This chapter contains the following sections:

- What Is Deployment?
- Phases of the Deployment Process

What Is Deployment?

In this book, *deployment* refers to the process of setting up Netscape Application Server (NAS) at your site. When installing and migrating NAS to your production environment, there are several issues to consider and steps to follow for the deployment process to fully take advantage of your particular corporation's environment and needs. This book is organized according to these recommended steps.

To further clarify, in this book *deployment* does not refer to deployment of applications. Application deployment is one of many issues involved in deploying NAS, but it is not by any means the only or most important issue. For details about deploying applications, see Chapter 2, “Deploying and Upgrading Applications,” in the *Administration Guide*.

Think of a successful deployment in terms of the following general goals:

Knowing your system’s capabilities A major goal of the deployment process is to develop a set of expectations for your NAS system and understand the system’s capabilities and limits. To better understand your server’s capacity, you need to determine such factors as number of users, typical peak capacity, typical session duration, types of requests being processed, and so on. You gather this information and then analyze it to establish a processing threshold for your site.

Maximizing performance Once you understand Netscape Application Server’s capabilities, you can focus on maximizing performance within your system’s processing scope. Your number one goal in deploying NAS is to make sure that its performance is optimal for your environment. Performance consists of two measurable factors: primarily throughput, and, in some instances response time.

Throughput refers to capacity, or the number of requests that NAS can service in a given time period. A request consists of a single user’s attempt to access data, and the return of that data by the server. Throughput is measured in requests per minute or requests per second.

Response time is the time it takes for a request to be processed. You begin measuring response time from the moment a user submits a request until the moment the server returns the result of the request back to the user. The unit of measurement for response time is seconds.

Improving scalability Scalability refers to the ability to support an increase in users and requests, also referred to as an increase in load, by proportionally increasing hardware resources, *without degrading response time or system stability*. In other words, scalability is the system’s ability to cope with an increase in users and requests when that increase is matched with a proportional increase in hardware resources. If response time degrades, despite the addition of hardware resources in proportion to the increase in users, then the system is not scalable.

To meet increased load, you can scale up the number of CPUs, the number of machines, or both. Throughput improves as you scale up your hardware resources.

Ensuring fault tolerance An important goal of deployment is to eliminate any single point of failure in your NAS system. If one or more servers fail, fault tolerance and fail-over capabilities ensure that transaction requests continue to be processed without interruption. The server fail-over capabilities and state and session management features of NAS can be configured to meet your ongoing processing needs.

Deployment Terminology

This guide uses a number of terms that relate to the process of deploying Netscape Application Server (NAS). Since these terms are often used by technology professionals in a variety of ways, they are listed and defined in the Glossary at the end of this book so that you understand their meaning in the context of NAS deployment.

Phases of the Deployment Process

For this book, the deployment process is divided into four phases, each one building on the previous one. By following these phases, you will address the deployment goals discussed earlier.

Planning Your Environment

Planning your environment is the first phase of deployment and consists of determining how Netscape Application Server (NAS) will fit into your overall enterprise. You may realize during this process that you must change the structure and components of your network to accommodate your NAS needs. Or, if your network structure is not flexible at this time, use the environment planning process to determine how you can best deploy NAS to fit within the existing network setup.

Determining System Capacity

Determining system capacity consists of figuring out what sort of load will be placed on your NAS system and how you can accommodate this load. During this phase, you'll develop a profile of the number of users accessing NAS at your site and the sorts of requests they submit. Then, based on this kind of information, you can decide how to structure and configure NAS so that it successfully handles the capacity requirements you expect to have.

Running Stress and Performance Tests

After you configure NAS, deploy a representative sampling of applications and run them over the system to determine Netscape Application Server's tolerance threshold. Wherever you identify bottlenecks, use this phase to fine-tune the system and improve performance.

Deploying the Production System

Finally, after you have tested and refined system performance, deploy your live NAS system. You may encounter new bottlenecks once you go into production, but at this point, you'll have experience fine-tuning the system and will know what sorts of mechanisms you can employ to improve performance.

Chapter

2

Planning Your Environment

This chapter explains how to fit Netscape Application Server (NAS) into your overall enterprise. Planning your environment is one of the first phases of deployment. You may realize during this process that you must change the structure and components of your network to accommodate your NAS performance needs. Or, if your network structure is not flexible at this time, use the environment planning process to determine how you can best deploy NAS to fit within the existing network setup while maximizing server performance.

This chapter contains the following sections:

- Establishing Performance Goals
- Assessing Your Overall Network Configuration
- Netscape Application Server Topology

Establishing Performance Goals

Before beginning any discussion on how to deploy Netscape Application Server (NAS), make sure you understand your performance goals and what you want to achieve when you integrate NAS into your enterprise. As explained in Chapter 1, “Overview of Netscape Application Server Deployment,” one of your main goals in deployment is to maximize performance. This translates into maximizing throughput and reducing response time. You will make decisions about your network configuration and how NAS fits into it based on your expectations of throughput and response time.

Maximizing Throughput

Throughput refers to capacity, or the number of requests per minute that your system can process. As explained in Chapter 1, “Overview of Netscape Application Server Deployment,” a request consists of a single user’s request for data, and the return of that data by the server. The request makes a round trip, from the user submitting the request, to the server, and then back from the server returning the result of that request to the user.

A simple example would be a shopping cart application, in which the user must click OK on a web page to submit a request to purchase an item. The result of this click, the purchase being processed, is considered a single request. Another example is a 401(k) application, where a user clicks OK to request a snapshot of an account balance, and the snapshot is returned to the user. A request is not a running application, but rather a transaction generated by an application to read or write data.

Improving throughput means increasing the number of requests per minute that can be handled by the server. A lower throughput capacity means that some or even many users are unable to process their transactions immediately, causing them to wait longer to obtain request results.

When planning your network environment and determining how NAS fits into your overall enterprise, consider what you can do to your network to increase the number of requests per minute that the system can handle.

Improving Response Time

Response time refers to the number of seconds it takes for request results to be returned to the user. Consider the 401(k) application example provided in “Maximizing Throughput.” When a user requests a 401(k) account balance on a web page, ideally the information should be displayed on the page within a few milliseconds from the moment the user clicks OK. However, if performance is not optimal, the user may have to wait several seconds, perhaps even minutes, for the account balance to appear. Response time is discussed in detail in Chapter 3, “Determining System Capacity,” but when considering how to integrate Netscape Application Server into your overall network, think about what you can do to your network to improve the average response time of user requests.

Assessing Your Overall Network Configuration

When planning how to integrate Netscape Application Server (NAS) into your network so that performance is maximized, consider the following areas:

- Assessing Network Components
- Preparing for Network Bottlenecks
- Planning Firewall Location

Assessing Network Components

Network components affect the performance of your NAS system. As you decide on the desired size and bandwidth of each, first determine your network traffic and identify its peak. See whether there is a particular hour, day of the week, or day of the month in which overall volume peaks, and then determine the duration of that peak. Consider the additional traffic that adding NAS to your overall network will generate. At all times consult network experts at your site about the size and type of all network components you are considering adding.

Internet Access Lines

In making decisions about how to accommodate or improve Internet access, remember that your primary goal should be to accommodate the traffic of as many packets of data as possible. User requests and the responses to them are bundled into packets that travel across Internet access lines. A variety of Internet access lines exist, and what you use at your site affects traffic. Some examples of access lines include T-1 at 1.544 Megabits per second (Mbps), T-3 at 44.184 Mbps, fiber optic at 10–100 Mbps, and ISDN at 128–150 Kilobits per second (Kbps). The more access line bandwidth you have on any of these, the greater the number of packets that can travel back and forth along your lines.

Plan for Peak Load Times

During peak load times, the number of packets that are being sent is at its highest level. In general, scale your system with the goal of handling 100 percent of peak volume. But bear in mind that any network behaves unpredictably and that despite your scaling efforts, 100 percent of peak volume cannot always be handled.

For example, assume that at peak load 5 percent of your users occasionally do not have immediate Internet access when running their NAS applications. Of that 5 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 the peak, start by verifying that your Internet service provider (ISP) has a backbone network connection that can reach an Internet hub without degradation.

Increase Bandwidth

Next, determine by how much you must increase bandwidth. Depending on your method of access (T-1 lines, ISDN, and so on), you can calculate the amount of increased capacity you require to handle your estimated load. For example, suppose your site uses T-1 or the higher-speed T-3 links for Internet access. Given their bandwidth, you can estimate how many lines you'll 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 both of these figures using any of the web-site-analysis and-site monitoring tools currently available on the market.

A single T-1 line can handle 1.544 Mbps. So 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 bits / 8 bits = 772,000 bytes per second

772,000 bytes per second / 30 KB = approximately 25 concurrent client requests for pages *per second*

At 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.

Accommodate for Peak Load

Keep in mind, however, that 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 it is. For example, in the scenario outlined here, if peak load lasts for two hours and takes up 30 percent of the total load of 2,160,000 pages, this means 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 must increase the number of T-1 lines from four to 16:

648,000 pages / 120 minutes = 5,400 pages per minute

5,400 pages per minute / 60 seconds = 90 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. Naturally, the other 70 percent of your load can be handled throughout the rest of the day by these many lines. Note that instead of installing 16 T-1 lines, you get more value by installing a single T-3 line, which at bandwidth of 44.184 Mbps is equivalent to 28 T-1 lines, but equivalent in cost to only eight T-1 lines.

As a general rule, given existing Internet access technology, you should design NAS applications that return HTML pages of no more than 30 KB each. In addition to improving overall performance, you can better calculate and predict traffic over Internet access lines if the maximum size of your output pages remains constant. And remember that as ADSL, ISDN and cable modem technology becomes more popular, you'll get better throughput and speed on your applications.

Routers and Subnets

Routers and subnets have an impact on load balancing in NAS. Chances are, your network is already divided into subnets using routers. Or, perhaps, as a result of deploying NAS, you are planning to split up your network into new subnets. For example, you may wish to cordon off certain servers because the databases they communicate with are used by a specific functional group, for instance accounting or customer service. Consider the effect of subnets and routers on the load-balancing capabilities of NAS.

Load Balancing and Subnets

NAS scalability is increased by dynamic load balancing, a feature that distributes user request loads across designated servers by selecting the “best” server to process incoming requests. The best server is one that is determined to be least loaded according to the load-balancing system’s server statistics. You can add servers as needed and balance loads so that all incoming user requests are processed immediately.

There are several advantages to load balancing:

- high performance of Internet applications even under heavy user loads
- continuous application availability using fail-over and failure recovery
- a sophisticated internal communications scheme that eliminates any single point of failure
- dynamic scalability—the ability to add servers to the load balancing group as needed, without interrupting application use by users in current sessions
- The Web Server Load Balancer plug-in, which eliminates the need for a “primary” NAS and hence eliminates any single point of failure in the system. The plug-in is able to locate the best server based on a response table that it builds dynamically during runtime.

If you intend to use load balancing, then all the servers that participate in the process must be in the same physical subnet. If they are not, you must enable multicasting across the subnets by enabling it on the router that connects the servers in the separate subnets. Consult your network router documentation and your network administrator for details about configuring subnets.

Network Cards

For greater bandwidth and optimal network performance, use NAS with 10 Mbps Ethernet cards or, preferably, 100 Mbps Ethernet cards.

Preparing for Network Bottlenecks

Being prepared for network bottlenecks can help you avoid problems with throughput and response time, especially during peak load times. To avoid bottlenecks, focus on the following areas:

Improve your Internet access lines. Make sure that you have enough bandwidth on the physical media used to carry packets to and from your web browser client and NAS, via the web server.

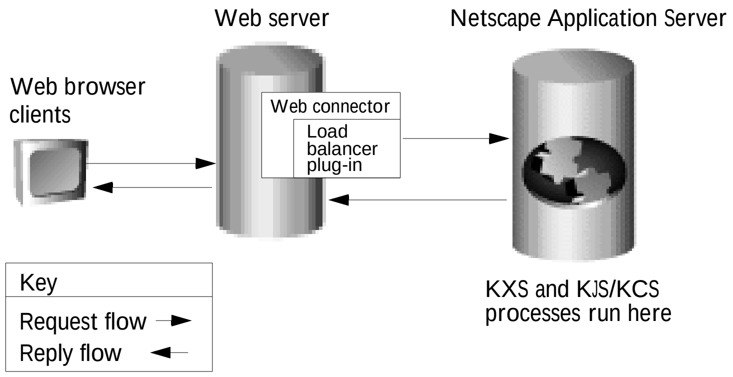
Reduce the number of routers. Routers can create bottlenecks, particularly when too many are configured into one network. Consult a network expert at your site about how to set up your network topology and avoid installing more routers than are absolutely necessary.

Reduce the number of subnets. One of the outcomes of having more routers is that your network becomes a composite of multiple subnets. For this reason, it's best to have as few subnets as possible. A high number of subnets results in a complicated network that packet traffic must travel on, sometimes negatively affecting response time.

Planning Firewall Location

Typically, you deploy NAS with a web server front end, such as Netscape Enterprise Server (NES). A user accesses NAS data by sending a client request from a web browser through a web server to the Executive Server (KXS) process running inside NAS. In turn, the KXS manages the request and sends data back to the client through the web server.

The communication between the web server and NAS is enabled by the Web Connector plug-in installed on the web server. If you are not using one of the NAS-recommended web servers, the communication is handled through a CGI process. Its is strongly recommended that you use NES.



Any entry into NAS from the Internet, or even from an intranet, exposes some or all of your network to a far-reaching audience that can go beyond your enterprise. To protect your network, it is strongly recommended that you deploy one or more firewalls. Typical firewall schemes are described in "Firewall Topologies" on page 29.

Where you place the firewall and how you configure it to allow connections to pass through depends on your security priorities and existing network framework. Consider two issues in particular:

- the protocols or communication processes that occur between NAS and the web server, and between NAS and the back-end data sources
- the firewall filters you should set up to ensure data integrity of the packets that are sent to and from NAS through the firewall

Communication Protocols

Two communication protocols exist between the web server and NAS–Transmission Control Protocol (TCP/IP) and User Datagram Protocol (UDP):

- **TCP/IP** handles a variety of data transfers between the web browser and Netscape Application Server. In addition, TCP/IP fragments packets when data transfer occurs and reassembles them when they arrive at their destination.
- **UDP**, a connectionless protocol within TCP/IP, is used to verify that the connection between the Web Connector plug-in and the NAS KXS engine is still active.

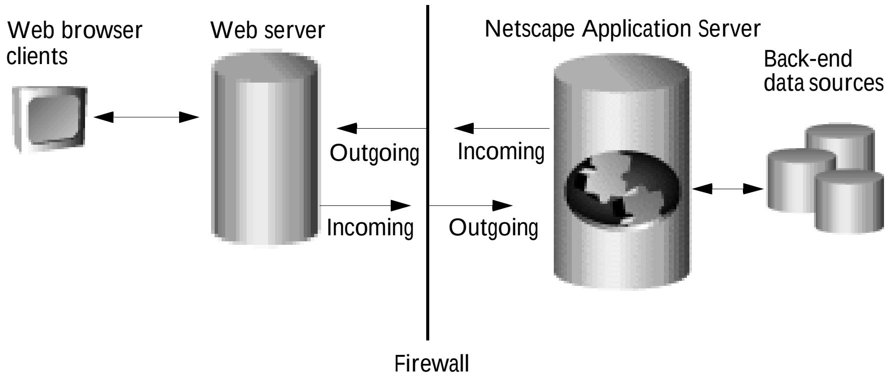
TCP/IP and UDP connections also exist between NAS installations, along with an IP Multicasting protocol connection, which is used for load balancing across the servers. Multicasting connections are always blocked on firewalls and can occur only through multicast-enabled routers.

Firewall Filters

The most common type of firewall technology is packet filtering. When a firewall is set up, connections into your network from the outside, and replies to these connections leaving the network, must pass through a packet filter. Think of a connection as a packet that has been sent over a network line. When it arrives at its destination, the connection has occurred. A reply to this packet is the response.

A packet filter examines the information contained in the header of each packet. Headers usually contain source and address data, such as address and port number, as well as the direction or flow of the packet. The flow of these connections and replies in and out of the firewall through these filters are as follows:

- incoming from the web server to the firewall
- outgoing from the firewall to NAS
- incoming from NAS to the firewall
- outgoing from the firewall to the web server



Initially, the filters are configured with the following defaults for enabling or blocking the connections and replies:

	Incoming from web server to firewall	Outgoing from firewall to NAS	Incoming from NAS to firewall	Outgoing from firewall to web server
Connection	Blocked	Blocked	Allowed	Allowed
Reply	Allowed	Allowed	Allowed	Allowed

Opening Ports to Allow Firewall Connections

To allow a particular connection or reply to pass through the firewall, you need to open up a port on the firewall. Each kind of connection, depending on the protocol used, requires opening up a specific port on the firewall.

The following table lists the connections and replies that pass through the firewall, the protocols used for each, and their default ports.

Connection	Protocol	Default port
Web Connector plug-in to KXS engine	TCP/IP	10818
Web Connector plug-in return connection between the web server and NAS	TCP/IP	Any port greater than 1024
Web Connector ping request from web server to the KXS engine	UDP	9610
Web Connector ping return from KXS engine to the web server	UDP	Any port greater than 32768
KXS load-balancing information from one KXS engine to another server's KXS engine	Multicasting	9607

Connection	Protocol	Default port
Administration information from one KAS engine to another server's KAS engine	Multicasting	9608
Netscape Application Server Administrator UI to KAS engine	TCP/IP	10817

Firewall Topologies

A topology is a schematic layout of your network. It is the logical map of your servers, hosts, clients, and other elements, and it shows the connections made between them. A firewall topology focuses on where the firewall exists within your network and how the rest of your enterprise interacts with it. The topologies described here are high-level, logical organizations of where the firewall falls within the system.

When deciding on a firewall type and where to place it, think about your security needs and existing network framework. For example, a firewall may already be in place at your site. If so, you should decide where you want to place NAS in relation to the existing firewall.

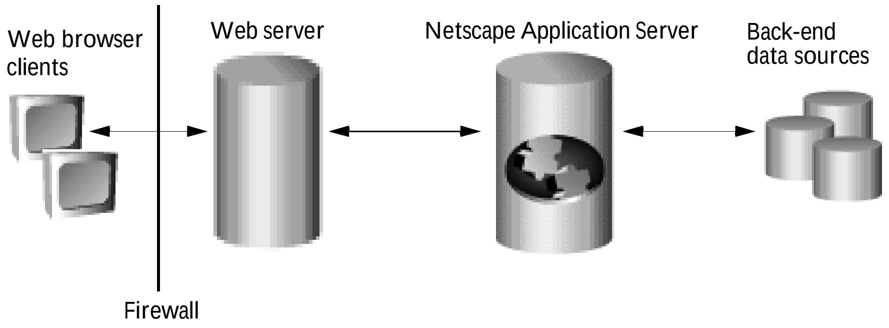
The following topologies are discussed in this section:

- Single firewall
- DMZ firewall
- DMZ-Database protection firewall

These topologies represent some of the more common ways that firewalls are configured within networks in which NAS is installed. No one topology is better than the others; however, one may make better sense for your deployment needs.

Single Firewall Topology

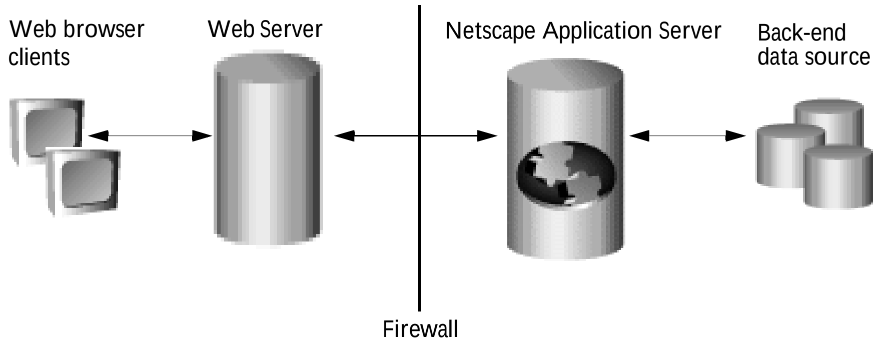
The most common firewall configuration is the single firewall, located either between the web server and the Internet, as shown next, or between the web server and NAS.



In this topology, you must configure the firewall in the following manner to allow HTTP requests to traverse through the firewall from the web browser client to NAS:

- Allow incoming HTTP requests from the web browser client to the web server by enabling port 80.
- Allow incoming HTTPS requests by enabling port 443.

When the firewall is located between the web server and NAS, as shown next, the web server is exposed to the Internet and is thus vulnerable to attacks.

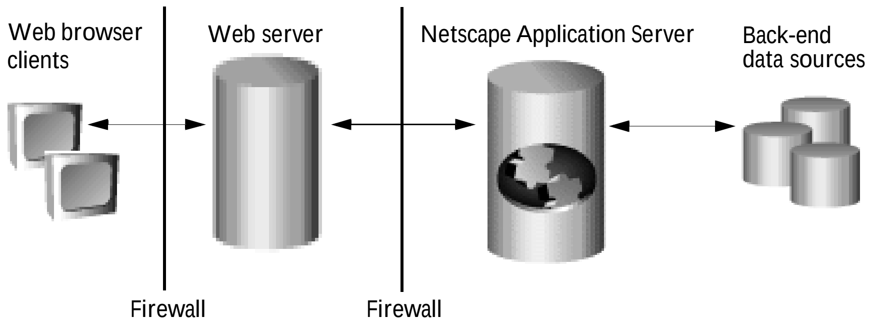


In this topology, you must configure the firewall in the following manner:

- Allow incoming TCP/IP requests from the web server to the KXS engine by enabling port 10818 on the firewall.
- Allow incoming UDP requests from the web server to the KXS engine by enabling port 9610 on the firewall.
- Allow outgoing UDP replies from the KXS engine to the web server by enabling a port greater than 32768.
- Allow outgoing replies from the KXS engine to the web server by enabling port 1024 or higher.

DMZ Firewall Topology

The increasingly popular DMZ ("demilitarized zone") topology is useful if you want to open up your private network to business partners and customers. It adds an extra layer of security beyond the single firewall layer. In this topology, not only do you get a double layer of security from two firewalls, you also gain from having additional monitoring activities in the area within the two firewalls.



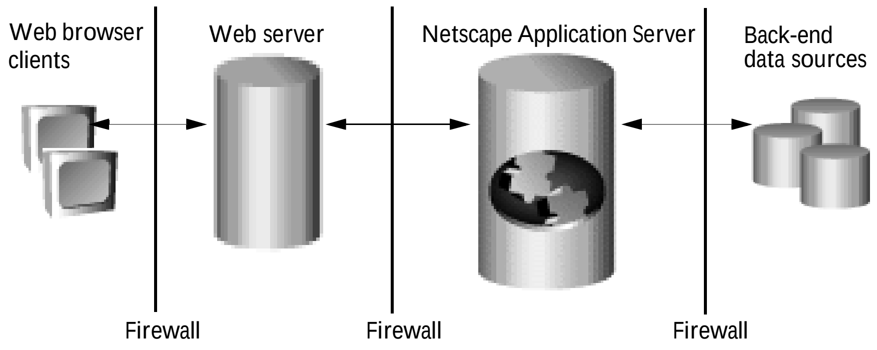
The outer firewall, that is, the one between the web browser and the web server, creates a public DMZ. The inner firewall, the one between the web server and Netscape Application Server, creates a private DMZ. You can set up a proxy server on this inner firewall that creates alias IP addresses of the NAS machines that are exposed, via the web server, to the web browser client.

In this topology, configure the *outer* firewall to allow incoming TCP/IP requests from the web client to the web server by enabling port 80.

Configure the *inner* firewall to allow incoming TCP/IP requests from KXS, outgoing TCP/IP replies from the KXS engine to the web server, incoming UDP requests from the web server to the KXS engine, and outgoing UDP replies from the KXS engine to the web server by enabling the various ports that allow these connections. See the previous section "Opening Ports to Allow Firewall Connections" for details.

DMZ–Database Protection Firewall Topology

On rare occasions, you may want to structure your enterprise so that corporate databases reside on their own subnets. For example, if you want to protect sensitive financial data, you may decide to separate into a subnet the database or databases used by the finance department. In addition to the deployment issues described earlier in "Routers and Subnets" on page 24, you should also consider deployment issues around setting up a firewall between the database subnet and the rest of your network. Consult your network administrator for details.



Netscape Application Server Topology

There are many ways you can set up your Netscape Application Server (NAS) system. This section describes some common, as well as recommended, configurations and contains the following topics:

- Examples of NAS Topologies
- Determining Which Topology to Use
- Determining Backup Requirements
- Integrating a Database Back End
- Integrating NAS with Directory Server

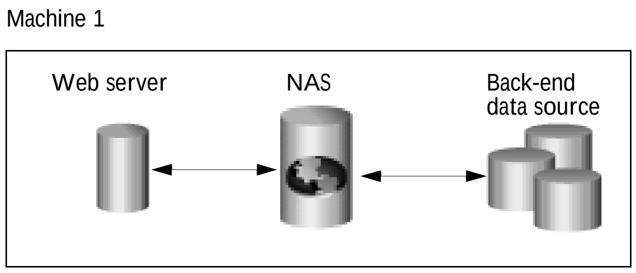
Examples of NAS Topologies

The following topologies range from a very simple scheme to the most complex. No single topology is best for all companies, since every enterprise has its own unique set of requirements and circumstances. Deciding which topology is best suited for your site is something you'll determine over time, as you fine-tune and continue to use NAS within your organization.

In all of the following topologies, the "back-end data source" represents any type of back end that provides and stores data. Some examples of a back end include a mainframe system, a database, another application server, a legacy application, and so on.

Topology 1: Single-Machine Configuration

The simplest NAS topology is one that consists of a single machine on which you have installed a web server, a Netscape Application Server, and a back-end data source.



This topology has the following advantages:

- It's a low-cost solution in a setting where there are likely to be few concurrent users, such as in a simple intranet.
- It's easy to install because all elements of the system are located on a single machine.
- It's easy to administer, again, because all elements of the system are located on a single machine.
- A minimal number of bottlenecks occur because the system is independent of the rest of the overall network. Likewise, this topology causes minimal bottlenecks on the rest of the network because the potential bottleneck areas (communications between the web server and NAS, and between NAS and the back-end data source) are physically located on one machine.

This topology has the following disadvantages:

- The web server, NAS, and back-end data source compete with one another for use of a single system's resources.
- Depending on the scalability of the machine, the number of concurrent users the system can support is reduced because of the contention for system resources. A high-end machine with multiple CPUs and high memory can obviously support more users than a lower-end machine.
- A single machine system has no fault tolerance: if for some reason the machine fails, the entire system shuts down without any failover provision.
- If you want to configure a firewall, and your data source has sensitive information you want to secure, you can only set up the firewall outside the web server as opposed to between NAS and the data source.

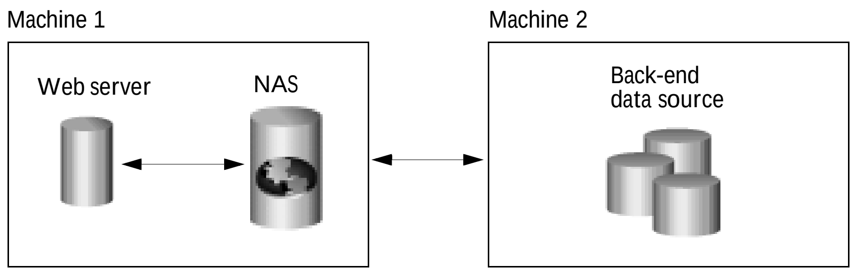
Table 2.1 summarizes deployment issues and how they rate for Topology 1. A high score (5 is the highest; 1 is the lowest) means that for this topology, the particular issue rates well for deployment. All the topologies described in this section are rated according to the same criteria so that you can easily compare and decide which topology is best suited for your enterprise.

Table 2.1 Ratings for Topology 1

Deployment issue	Score	Details
System administration	3	
	5	Ease of daily administrative tasks
	1	Ease of troubleshooting system-level problems
Hardware/OS resource usage	1	
	1	Competition for memory
	1	Competition for I/O
	1	Competition for CPU
Availability	2	
	1	System availability during regular maintenance
	1	System availability during machine-level failure
	5	System availability during process-level failure
Impact on existing (legacy) environment	3	
	3	Impact on stability of legacy systems and applications
	3	Impact on performance of legacy systems and applications

Topology 2: Two Machines

Another simple topology that improves on performance by easing the load is one in which the web server and a single installation of NAS reside on the same machine, while the back-end data source is on another machine.



As with Topology 1, this topology is adequate in a setting with few concurrent users, such as in a simple intranet. The advantages and disadvantages of this topology are similar to Topology 1, with the exception of the following three points:

- A two-machine topology means that resources are distributed more generously. The back-end data source has dedicated memory, I/O, and CPU resources on its machine, thus freeing up more of these resources for NAS and the web server on the machine they share.
- If you want to configure a firewall and your data source has sensitive information you want to secure, you can set up a firewall outside the web server as well as between NAS and the data source.
- This topology is not as independent of network traffic as Topology 1 because two machines are involved. Therefore, a potential network bottleneck could occur between NAS and the back-end data source.

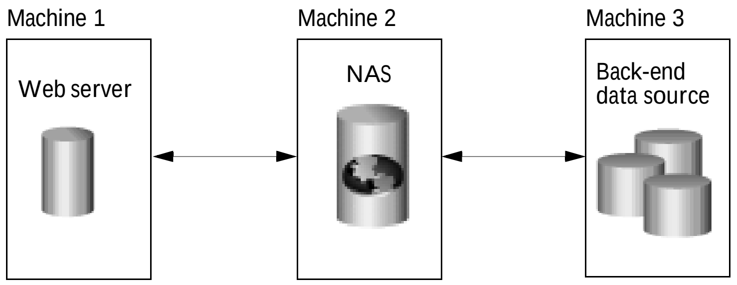
Table 2.2 summarizes deployment issues and how they rate for Topology 2. A high score (5 is the highest; 1 is the lowest) means that for this topology, the particular issue rates well for deployment. All the topologies described in this section are rated according to the same criteria so that you can easily compare and decide which topology is best suited for your enterprise.

Table 2.2 Ratings for Topology 2

Deployment issue	Score		Details
System administration	4		
		5	Ease of daily administrative tasks
		3	Ease of troubleshooting system-level problems
Hardware/OS resource usage	3		
		3	Competition for memory
		3	Competition for I/O
		3	Competition for CPU
Availability	2		
		1	System availability during regular maintenance
		1	System availability during machine-level failure
		5	System availability during process-level failure
Impact on existing (legacy) environment	4		
		4	Impact on stability of legacy systems and applications
		4	Impact on performance of legacy systems and applications

Topology 3: Three Machines

The following topology consists of three separate machines: the web server is installed on one machine, a single installation of NAS is installed on the second machine, and the back-end data source is installed on the third machine.



This topology lends itself to more accurately analyzing performance and identifying bottlenecks as each of the components resides on a separate machine. If performance problems arise, you can more easily isolate them and determine where the bottleneck exists.

One or more web servers reside on separate machines and thus can handle more multiple concurrent requests than Topologies 1 and 2. In this topology, if the web server is to be used strictly for front-ending NAS requests, make a point of installing the web server on a low-end machine, since the kind of processing the web server performs is simple traffic control of requests. This frees up your more powerful machine for the NAS installation, which typically requires far more resources than the web server.

In this topology, if you find that the web server cannot handle the required number of concurrent requests, it is recommended that you run more web server instances on the web server machine. Note that, via the MaxProcs registry setting, Unix supports multiple web server instances running on a single machine, whereas NT does not. On NT, if you intend to have more than one web server instance, install each web server on a separate NT system.

As long as a set of web server installations or instances are serving a single web site, you can also improve web server load by placing a web server load balancer, such as Cisco's LocalDirector, between the client browser and the web servers or instances. The purpose of such a load balancer is to distribute the request load evenly among the web servers or web server instances.

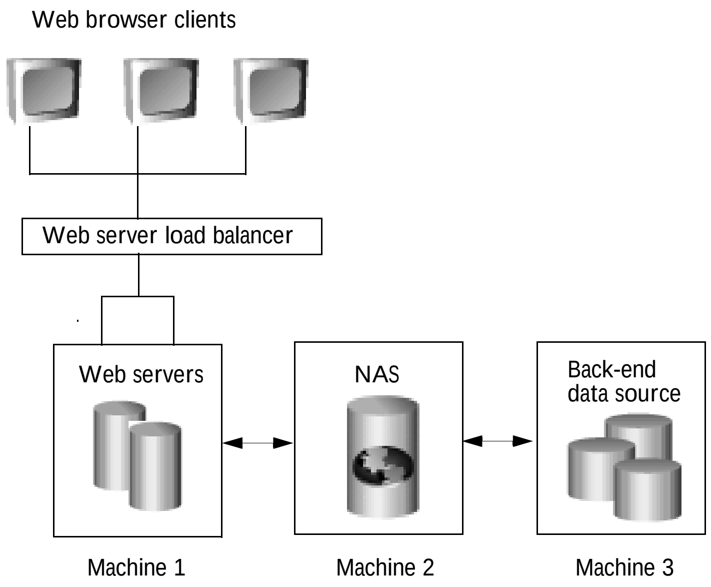


Table 2.3 summarizes deployment issues and how they rate for Topology 3. A high score (5 is the highest; 1 is the lowest) means that for this topology, the particular issue rates well for deployment. All the topologies described in this section are rated according to the same criteria so that you can easily compare and decide which topology is best suited for your enterprise.

Table 2.3 Ratings for Topology 3

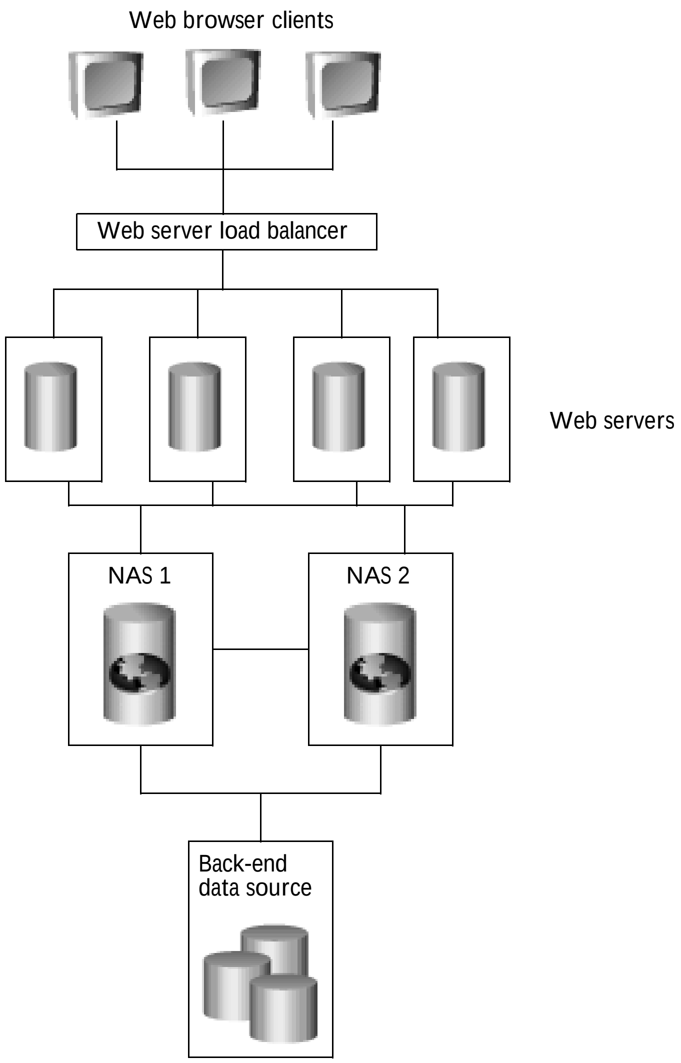
Deployment issue	Score		Details
System administration	5		
		5	Ease of daily administrative tasks
		5	Ease of troubleshooting system-level problems
Hardware/OS resource usage	5		
		5	Competition for memory
		5	Competition for I/O
		5	Competition for CPU
Availability	2		

Table 2.3 Ratings for Topology 3

Deployment issue	Score	Details
	1	System availability during regular maintenance
	1	System availability during machine-level failure
	5	System availability during process-level failure
Impact on existing (legacy) environment	5	
	5	Impact on stability of legacy systems and applications
	5	Impact on performance of legacy systems and applications

Topology 4: Scaling to Two NAS Machines

In addition to increasing web server resources, you may also want to increase NAS resources to handle your system's throughput. By adding another NAS machine, the two systems can form a cluster (shown here), which is a group of NAS installations that participate in synchronization of state and session data.



Topology 4 has the following advantages:

- With multiple NAS machines, you can achieve fault tolerance (if one machine goes down, the other one remains to continue processing).
- A second machine provides the ability to scale across more CPUs, memory, and I/O resources.

This topology can be further augmented by dedicating multiple web servers for each application server. It is recommended that you upgrade to this topology if stress testing proves that Topology 3, described on page 38, is inappropriate for your processing needs. Using this variation of the topology also depends on whether or not the web server is dedicated to serving NAS only or static and dynamic transactions of its own.

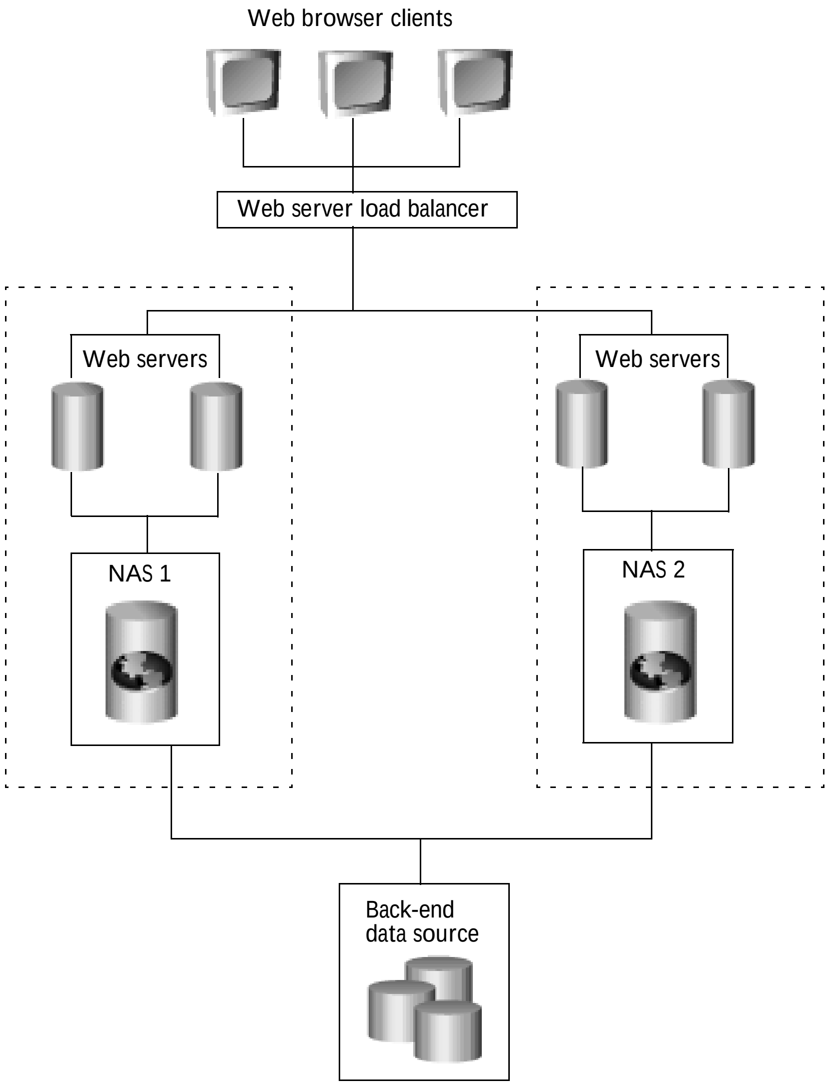


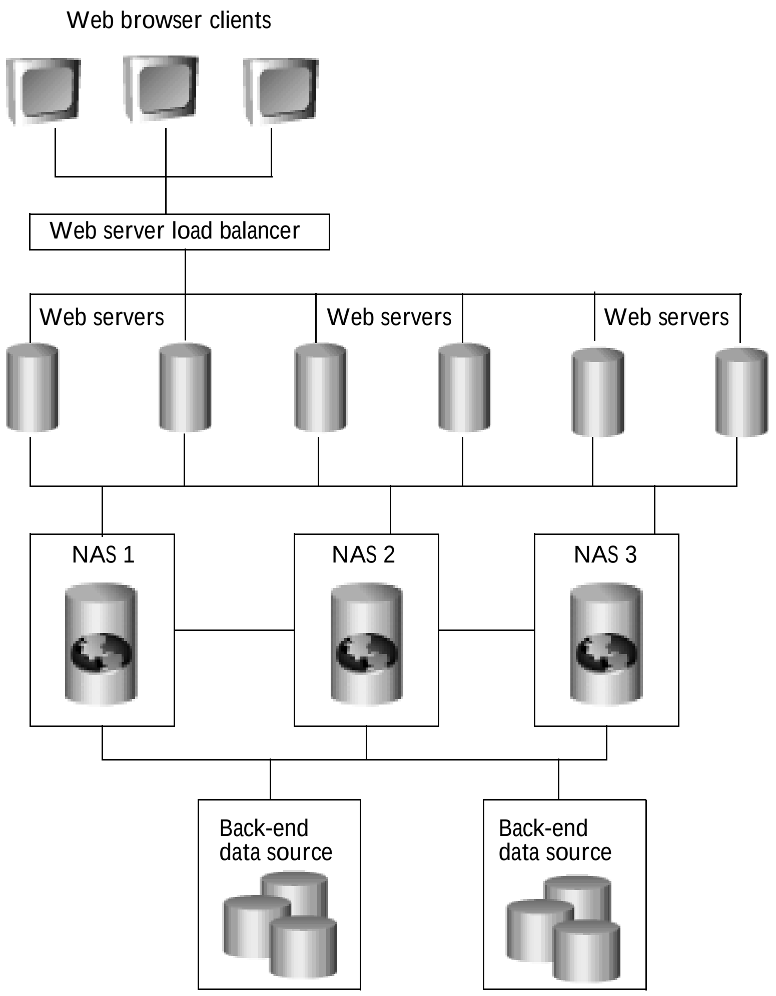
Table 2.4 summarizes deployment issues and how they rate for Topology 4. A high score (5 is the highest; 1 is the lowest) means that for this topology, the particular issue rates well for deployment. All the topologies described in this section are rated according to the same criteria so that you can easily compare and decide which topology is best suited for your enterprise.

Table 2.4 Ratings for Topology 4

Deployment issue	Score		Details
System administration	5		
		4	Ease of daily administrative tasks
		5	Ease of troubleshooting system-level problems
Hardware/OS resource usage	5		
		5	Competition for memory
		5	Competition for I/O
		5	Competition for CPU
Availability	4		
		4	System availability during regular maintenance
		4	System availability during machine-level failure
		5	System availability during process-level failure
Impact on existing (legacy) environment	5		
		5	Impact on stability of legacy systems and applications
		5	Impact on performance of legacy systems and applications

Topology 5: Scaling to More Than Two NAS Machines

The final topology is one in which the cluster consists of more than two Netscape Application Servers. This configuration is typically accompanied by a proportional increase in the number of web servers, as shown here.



There are two important reasons for increasing the number of NAS servers beyond two:

- To achieve greater fault tolerance
- To meet higher scalability and performance requirements

Do not assume that scaling up the number of NAS servers in a cluster results in a linear scaling of throughput. Some of the resources of each machine are used to maintain load balancing and state and session management information across the various servers. Adding more servers does improve throughput, but each additional server introduces more overhead in the form of load balancing and data synchronization. This overhead is referred to as the scaling factor. The added servers must devote resources to the scaling factor as well as to the goal of improving performance.

Table 2.5 summarizes deployment issues and how they rate for Topology 5. A high score (5 is the highest, 1 is the lowest) means that for this topology, the particular issue rates well for deployment. All the topologies described in this section are rated according to the same criteria, so that you can easily compare and decide which topology is best suited for your enterprise.

Table 2.5 Ratings for Topology 5

Deployment issue	Score		Details
System administration	5		
		4	Ease of daily administrative tasks
		5	Ease of troubleshooting system-level problems
Hardware/OS resource usage	5		
		5	Competition for memory
		5	Competition for I/O
		5	Competition for CPU
Availability	5		
		5	System availability during regular maintenance
		5	System availability during machine-level failure

Table 2.5 Ratings for Topology 5

Deployment issue	Score		Details
		5	System availability during process-level failure
Impact on existing (legacy) environment	5		
		5	Impact on stability of legacy systems and applications
		5	Impact on performance of legacy systems and applications

Determining Which Topology to Use

You should test different NAS configurations and machine-CPU mixes to determine which combination works best for you. Naturally, performance, particularly response time, is also affected by the nature of your applications and what kinds of processing power they require. For example, do the applications involve simple I/O activity, or are complicated calculations used? This and other application capacity issues are explained in detail in Chapter 3, “Determining System Capacity” and in Chapter 4, “Performance Testing and Fine-Tuning Your System.”

Topologies 4 and 5 consist of more than one installation of NAS on your network. You can choose one of these topologies as a way of handling greater loads, or you can elect to scale the number of CPUs within each machine.

A single-machine topology (Topology 1) may not perform as well as you would like. If your main concern is to maximize throughput, then as a general rule of thumb, adding a second machine with NAS installed on it is a desirable solution. A second machine can handle a larger volume of user requests than a single machine can. However, if your main concern is response time, then consider adding CPUs to the machine or machines you already have. Having more CPUs speeds processing. But remember that system performance is also related to application design, so these suggestions will help improve performance in varying degrees, depending on how you design and deploy your applications.

Determining Backup Requirements

Your server backup requirements may affect the number of NAS installations at your site. Your server backup objective is to achieve fault tolerance, eliminating any single point of failure in your NAS system. If one or more servers fail, fault tolerance and fail-over capabilities ensure that requests continue to be processed without interruption. This is achieved through data synchronization, which enables distributed state and session management services among servers in the same cluster. Distributed state and session management services preserve data generated during a user session so that user sessions continue without interruption, and with no loss of data, even if one or more servers or processes in the cluster become unavailable.

A cluster is a group of NAS machines that participate as a group in synchronization of state and session data. Each server within a cluster can assume one of several Sync Server roles:

- Sync Primary—The primary data store, to which all other servers in a cluster communicate for the latest distributed data information.
- Sync Backup—The server that mirrors the information on the Sync Primary and takes over the role of the Sync Primary if the original Sync Primary fails.
- Sync Alternate—A server that is eligible to become a Sync Backup. If the number of Sync Backups falls below the set maximum (always 0 or 1), the Sync Alternate with the highest priority relative to other Sync Alternates is promoted to Sync Backup.

If your configuration consists of only one NAS machine, then cluster planning is not necessary. However, if you want to perform data synchronization, you must plan your cluster carefully and consider scaling the number of machines you deploy according to how you want to structure the cluster.

For information about how to assign a Sync Server role to one of the servers in your cluster, see Chapter 14, “Managing Distributed Data Synchronization,” in the *Administration Guide*.

Adding a Sync Backup to the Cluster

Adding another machine as a Sync Backup ensures that you always have a copy of your data, regardless of what happens to your Sync Primary; but it also means that overall performance is affected by the added load of data synchronization processing. Every change in the state of the Sync Primary must be replicated on the Sync Backup. This replication can significantly increase network traffic and memory allocation and deallocation. Therefore, do not use a Sync Backup unless it's absolutely necessary.

For any cluster, the Sync Server role of each server is dependent on your Data Synchronization needs. For additional scalability, increase the number of CPUs within a machine, or increase the number of machines. Another option, if you want to increase the number of machines, is to add another cluster that contains the additional machines. Your course of action will depend on the demands your application makes on your system.

For more information about failover capacity planning, see Chapter 3, "Determining System Capacity."

Integrating a Database Back End

Database integration and connectivity are two very important areas of NAS deployment. There are two main issues to consider:

- avoiding database bottlenecks
- deciding between using global transactions or local transactions

Avoiding Database Bottlenecks

When integrating a database back end into your overall NAS topology, make sure that the database does not become the system bottleneck. Talk to your database administrator and discuss the feasibility of performing the following performance-enhancing steps:

- Make sure that the database is installed on a machine that can support the expected load.
- Make sure the database resides on the same subnet as NAS.

- Study your database's data access methods and determine what sorts of actions you can take on the database side to improve data access performance.
- Study the way your data is stored and determine if the data storage structure is optimal. This requires having a solid understanding of your database, something your database administrator can provide.
- If you have more than one NAS server, make sure there are multiple connections from NAS to your databases. If the connection to your database or databases is via only one of the NAS servers and that server fails for some reason, you lose your database connection.

Using Global Transactions or Local Transactions

If your NAS system has a variety of database back ends (Oracle, SQL, Sybase, and DB2), consider designing your Java-based applications to work with global transactions, which can update multiple database types. Also, if your NAS system integrates databases that are in different locations, global transactions can update these distributed databases.

Global Transactions

A global transaction updates a database using one or more Enterprise Java Beans (EJBs) running concurrently for the same global transaction, from within one or more KJS processes. Multiple EJB processing occurs when an EJB triggers another EJB to run and they both participate in the same transaction. A global transaction also updates multiple databases, of different types (Oracle, Sybase, and so on) that are distributed over different geographic locations. A local transaction, on the other hand, is managed not by an external transaction manager, but rather by a database itself.

As part of the deployment process, decide if you want to design your EJB applications to use global transactions for databases. Global transactions, also called distributed transactions, are managed by an external transaction manager, a feature you can enable at installation time or later by updating the NAS registry. For details about how to configure transaction manager at installation time, see the *Installation Guide*. For information about how to maintain and use transaction manager, see Chapter 10, "Administering Transactions," in the *Administration Guide*.

If you decide to use global transactions for certain databases (for example, those of Oracle and Sybase), keep in mind that you are allowed one connection per thread. Make sure that the maximum number of threads and database connections in your KJS processes are equal. Likewise, make sure that your database is configured to allow as many connections as the total threads and database connections of all your KJS processes.

For information about how to increase the number of threads and processes, see Chapter 7, “Increasing Fault Tolerance and Server Resources,” in the *Administration Guide*.

Local Transactions

Local transactions handle threads differently: here you can specify a connection pool of an unlimited size, causing performance at times to be better than with global transactions. Depending on the nature of your applications, you may decide that enabling local transactions is a better option.

Deciding Between Global and Local Transactions

Here are some application factors to consider when deciding between global and local transactions:

- the number of database requests that your applications generate
- whether the requests generate multiple calls to databases and then return the data after all calls have been made, or whether each call requires data to be returned before the next call is made
- the number of connections generated by your applications
- the number of resource managers required by your applications

Depending on all of these factors and your database structure and integration with NAS, you may decide to use a mix of both local and global transactions, if your database client supports this. Consult with your database administrator before making any decisions.

Before using transaction manager and resource manager, configure your database back ends for XA transactions. Consult your database vendor documentation for details.

Integrating NAS with Directory Server

The NAS installation program allows you to install Directory Server, Netscape's implementation of the Lightweight Directory Access Protocol (LDAP). NAS uses Directory Server to store NAS configuration information, most of which was stored in the registry of earlier NAS releases, and also uses it as a central repository for user and group information. Integrating NAS with Directory Server is particularly useful if you are installing multiple NAS machines at your site, because the configuration information for all your NAS installations is centralized in one place rather than being distributed across the registries of each NAS installation.

LDAP provides an open directory access protocol running over TCP/IP. Netscape Directory Server 4.0 supports LDAP versions 2 and 3, and provides the software necessary for an entire directory solution. It includes Directory Server, which is the server-side software that implements the LDAP protocol. Other LDAP clients are also available. These include the Users and Groups area in Netscape Administration Server and the addressbook feature in Netscape Communicator 4.0.

NAS Settings Stored on Directory Server

When you install Directory Server, many of the NAS configuration settings are stored on Directory Server. If you are upgrading from an earlier version of NAS, remember that all of these settings were previously stored on the NAS registry of each NAS installation at your site.

The NAS configuration settings that now reside on Directory Server are listed in Table 2.6.

Table 2.6 NAS configuration settings residing on Directory Server

Configuration Setting	Definition
ClassDef	All the registered applications that all NAS installations at your site use
NameTrans	The list of user-specified names for all applications registered to all your NAS servers and their corresponding GUIDs (globally unique identifiers).
Clusters	All the clusters you've created on your network and the servers within each cluster.

Table 2.6 NAS configuration settings residing on Directory Server

Configuration Setting	Definition
ACL	The list of access control lists (ACLs) that you can use to perform access checks to each application resource
EJB-Components	The list of user-specified names for all Enterprise Java Beans (EJBs) registered to all your NAS servers and their corresponding GUIDs (globally unique identifiers)
GMS	The Global Message Service (GMS) multicasting parameters. The load balancer module in each server communicates with load balancer modules in other servers using these multicasting messages.
NLS	The international environment settings for National Language Support (NLS). Used for developing single/ multi lingual applications with legacy character sets or Unicode character sets. Use this flag to enable or disable the NLS application programming interfaces (APIs).
Principal	The user and group security information for all installed NAS machines
DAE\DataSources	The mapping of data-source names to drivers
DAE2\DataSources	(JDBC) The mapping of data-source names to drivers
EB	The settings that control how enterprise beans are handled across all your NAS installations
Extensions	The extensions that are loaded into all your NAS installations when the servers are started up
LoadB	The load balancing parameters that control how requests are handled across all your NAS installations
REQ	The request manager settings used to configure threads in the thread pool
Security	The encryption parameters that control encryption between your web servers and your NAS installations

NAS-Directory Server Deployment Considerations

In addition to knowing what areas of NAS you can configure from Directory Server, note the following deployment tasks:

- If you have multiple instances of Directory Server installed at your site, and you wish to take advantage of Directory Server failover capabilities, keep the NAS configuration settings that are listed in Table 2.6 on one machine only, preferably on the configuration Directory Server as opposed to the storage Directory Server. The configuration Directory Server contains the `o=NetscapeRoot` tree, where all the information about your Netscape servers is stored.
- Multiple NAS installations can store their configuration settings on the configuration Directory Server. To avoid your settings being overwritten by another NAS installation's configuration settings, you are asked during NAS installation to assign a unique global configuration name to your NAS installation's settings. This name appears in the `o=NetscapeRoot` tree along with the global configuration names of other NAS installations. If you want to share configuration settings with other NAS installations, simply enter the same global configuration name for each installation.
- If you install Directory Server when you install NAS, specify that installation, rather than one that was previously installed at your site, as the configuration Directory Server and the storage Directory Server.
- When you install NAS, make sure that the Directory Server you point to has already been set up as a configuration Directory Server. In other words, the server you point to must contain the `o=NetscapeRoot` tree used by all your Netscape servers.
- When integrating Directory Server into your NAS system, you can install it on a machine that is dedicated to other non-Directory Server tasks. For example, you can install Directory Server on the same machine as NAS or on the same machine as your web server. The information that NAS accesses from Directory Server is cached in NAS, so the number of requests from NAS to Directory Server are minimal and do not require significant processing loads on the Directory Server machine.
- Only one Directory Server per NAS cluster is required. A single Directory Server can support multiple NAS clusters.

- For failover to work properly in Directory Server, install at least two instances—a master server and a slave—of Directory Server. This ensures that when the master (supplier) server fails, processing is redirected to the slave (consumer) server. The best type of replication is the supplier-initiated method, rather than consumer-initiated. It is recommended that you replicate Directory Server changes instantly, particularly because the changes that are generated by NAS to the Directory Server are likely to occur infrequently. It is therefore better to ensure that these few changes be fully propagated for load balancing purposes. This increases network traffic load, so it is important to find an optimal balance. Use the “Always keep directories in sync” option in the supplier-initiated replication process to ensure instant replication of NAS-related changes on Directory Server.

Read the *Installation Guide* for Netscape Directory Server 4.0 and the *Installation Guide* for Netscape Application Server 4.0 for details about LDAP issues.

Chapter

3

Determining System Capacity

This chapter explains how to measure Netscape Application Server (NAS) performance by determining system capacity. Determining system capacity is critical to a successful deployment of NAS. In this phase of deployment, you gather information about the level of activity that will occur on your server: you determine the number of users, the number of requests, expected response time, and expected application load.

Understanding system capacity is closely linked to understanding how NAS integrates into your current environment, a topic discussed in Chapter 2, “Planning Your Environment.” You may find that certain decisions you made earlier are directly linked to system capacity. Read the current chapter carefully, and then, if necessary, revisit the information presented in Chapter 2, “Planning Your Environment,” about planning your system topology. You may decide to adjust some of the decisions you made about NAS setup based on system capacity needs.

This chapter contains the following sections:

- Establishing System Performance Goals
- Developing a Capacity Profile
- Developing an Application Profile
- Developing a Failover Provision

Establishing System Performance Goals

At some point during the deployment process, decide what your system capacity goals are. Beyond the basic goals of improving response time and maximizing throughput, as explained in Chapter 2, “Planning Your Environment,” you should establish specific goals.

To help you plan system capacity, you should develop a profile of your system’s performance by determining the following information:

- 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 any three of these four pieces of information, you can always calculate the fourth.

Individual 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. Some examples of benchmarking tools include Benchmark Factory and WebBench. 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. For information about running performance tests with the help of benchmarking tools, see Chapter 4, “Performance Testing and Fine-Tuning Your System.”

Note that the results of calculations described in the following section, “Developing a Capacity Profile,” are not necessarily absolute. Treat them as reference points or benchmarks to work against as you try to fine-tune your server performance.

Developing a Capacity Profile

As your enterprise's NAS deployer, you have the responsibility of understanding your system and all its elements, including the applications you develop. Your system consists of a variety of elements, including the operating system, the network, web servers, back-end data sources, NAS servers, and your applications. The weakest link in this group will always be the bottleneck, so you need to pay attention to all of them when trying to improve capacity. You start by assessing the system's overall capacity and then identify where performance is being compromised.

This section explains how to assess overall system capacity by describing in detail the concepts of maximum number of concurrent users, peak load and peak capacity, steady state, response time, and requests per minute.

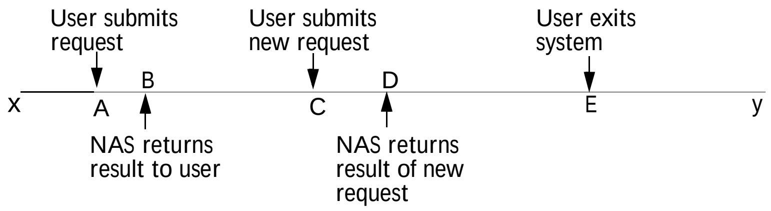
Defining Maximum Number of Concurrent Users

To better understand capacity, determine the maximum number of concurrent users that your system can handle before performance starts to decline. A user constitutes one running web-browser process that periodically sends requests from a client machine via a web server to NAS. A user is "concurrent" for as long as he or she is on the system as a running web-browser process submitting requests, receiving results of requests from the server, and viewing the results of the requests.

In the following diagram, think of the horizontal line (points x to y) as system activity. A single user is a concurrent user from point A until point E. From points A to B the user's request is being processed by the system.

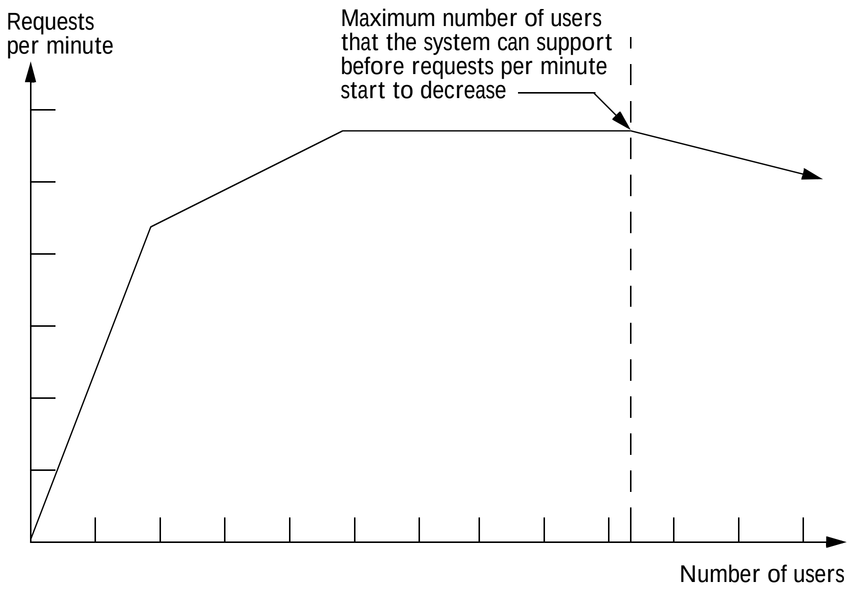
From points B to C the user is in think time mode, viewing the results of the request and deciding what action to take next. Realistically, a user does not submit requests continuously. A user submits a request, the server receives the request, processes it and then returns a result, at which point the user spends some time analyzing the result before submitting a new request. This time spent reviewing the result of a request is called "think time." The user is not performing any actions against the server, and thus a request is not being generated during think time. However, when you calculate the number of concurrent users, you include think time in the equation.

From points C to D, the system is again processing the request, with D being the point at which the result is returned to the user. From points D to E, the user is again in think time mode. At point E, the user exits the system, after which he or she is no longer a concurrent user.



Determining typical think time length is important because you can use it to calculate more accurately the number of requests per minute and 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 also means that you can support more concurrent users.

Eventually, in the course of system activity, as the number of concurrent users submitting requests increases, requests processed per minute begin to decline. The following diagram illustrates this situation:



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

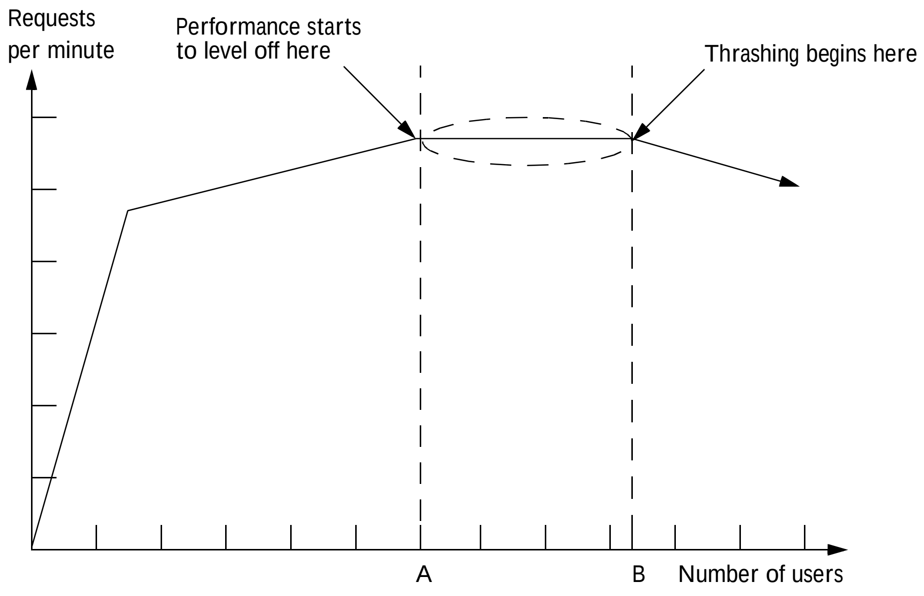
Defining Peak Load and Peak Capacity

Peak load refers to the maximum number of concurrent users and requests per minute that you expect the system to support, based on the pattern of activity that typically exists on your system. You may want your system to support a certain number of requests with a certain response time per request. This doesn’t mean that your system can in fact handle this load: it might be able to handle the number of users, but only by processing their requests at a slower rate.

If you decide you want your system to handle the maximum number of concurrent users that typically access the system and submit requests, you may have to add server resources to achieve the desired response time. You may, instead, decide to use another design point, steady state load, which is explained on page 61. Whether you choose peak load or steady state load as your system capacity design point depends on several factors, including acceptable response time, all of which are described later in this chapter.

Peak capacity refers to the maximum number of concurrent users that the system can realistically sustain before requests per minute start to decline and response time starts to increase. Peak capacity may be more or less than peak load, and knowing the difference is critical to planning your system. You may find out at first that your system cannot handle the peak load. However, if you use peak load as a design point, you can do things to adjust aspects of your system and bring peak capacity to a level that is acceptable for your enterprise.

In any system, as the number of concurrent users rises, so does the number of requests per minute (or per second). Then, at a certain point, although more users are concurrently on the system submitting requests, the number of requests per minute starts to level off and remain constant, as the following diagram illustrates:



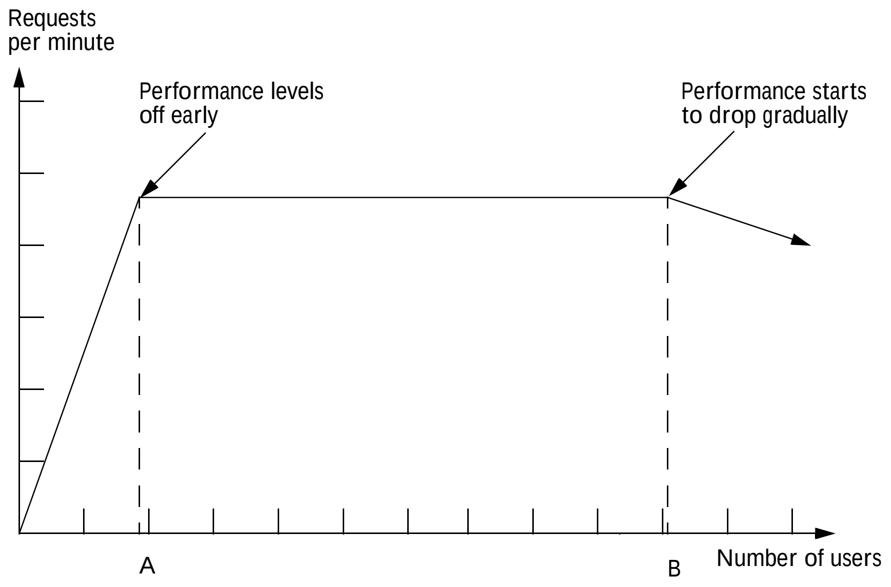
Point A represents the maximum number of concurrent users the system can handle before requests per minute start to level off. Point B represents peak capacity—the maximum number of users the system can support before performance starts to degrade. (Note that peak load is not represented in the diagram because it is not a function of the NAS performance curve, but rather a value defined by your system requirements.) Eventually, requests per minute start to decline, even though the number of concurrent users continues to rise. As the diagram above illustrates, this results in multiple users on the system but fewer requests being processed, a situation referred to as “thrashing.” Thrashing occurs when the system is spending more time and resources managing queues and loads than actually processing requests.

Defining Steady State

Steady state refers to the point at which the system reaches and maintains steady capacity, or a steady number of requests per minute, while the number of concurrent users continues to increase. In steady-state mode, your system is not processing at peak capacity; it is maintaining a steady capacity and continues to maintain this capacity, *even as the number of concurrent users on the system rises.*

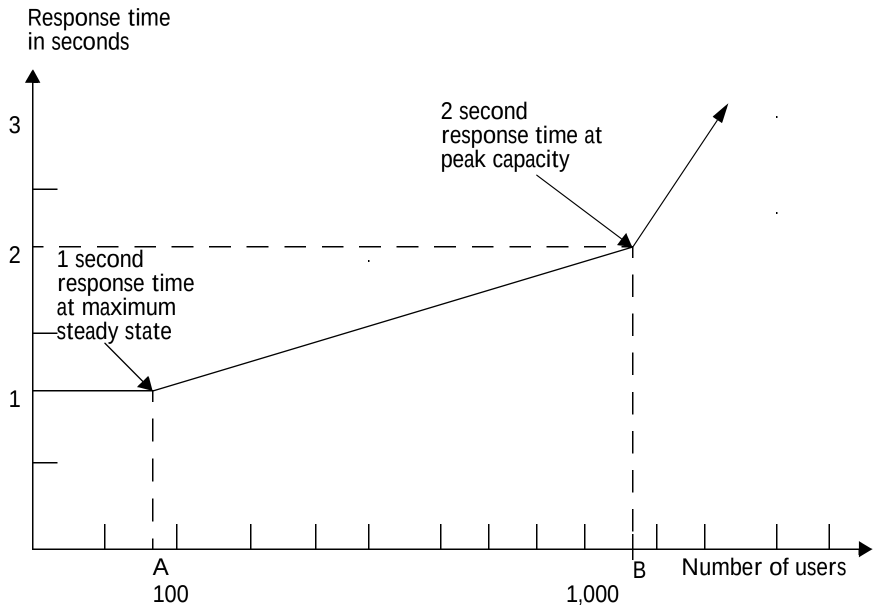
In the ideal system, capacity levels off with a relatively low number of users, meaning that the system can handle a large number of requests submitted by a small number of users. The system’s efficiency is further demonstrated by this large volume of requests per minute remaining at a steady state despite a continued increase in the number of concurrent users.

Here is how the ideal system looks graphically.



There is an initial steep slope upward in the number of requests per minute on the y-axis, to point A number of users on the x-axis. The distance and angle from point A to point B represents steady state: a steady increase in the number of concurrent users, without any change in the number of requests per minute. At point B, which represents peak capacity, 100 percent of system resources are being used, and any additional concurrent users causes requests per minute to decline. Ideally, when thrashing finally does occur at point B, the point at which system resources begin to be less efficient due to the number of users, the decline in requests per minute is gradual, as opposed to steep. Again, this graceful degradation in requests per minute indicates an efficient system: requests per minute decline slowly despite the continued increase in the number of concurrent users. Note that you should not design for any points past point B, as this represents an area of performance beyond capacity limits in which the system may react in an unpredictable manner.

Another way of analyzing steady state is in terms of response time, as you can see in the next graph.



During steady state, despite the added load of concurrent users, response time remains constant. For example, assume that the diagram represents a system in which steady state typically ranges from 0 to 100 concurrent users. Response time remains steady, despite the added load of users on the system. From point A to point B (1000 users), response time increases gradually. But after point B, which represents peak capacity, response time rises more dramatically as peak load is reached.

Designing for Peak Load or Steady State

When deciding your capacity goals, look at two factors: peak load and steady state. If your design point is peak load, then you must deploy a system that can sustain the expected maximum load of users and requests without a degradation in response time. This means that your system can handle the extreme cases of expected system load.

If your design point is steady state, then you don't have to deploy a system with all the resources required to handle the sever's expected peak load, or even the peak capacity; however, a system designed to support up to steady state load will have slower response time when peak load occurs.

Importance of Response Time

Remember peak capacity may not be able to meet the demands of the desired peak load. However, even at peak capacity, the response time may be greater than the acceptable response time limit that you get at steady state. If 3 seconds is your limit or threshold, but the system at peak capacity processes requests at a response time greater than 3 seconds, then you must determine which areas of your system you'll adjust so as to reduce response time to the desired 3 second limit.

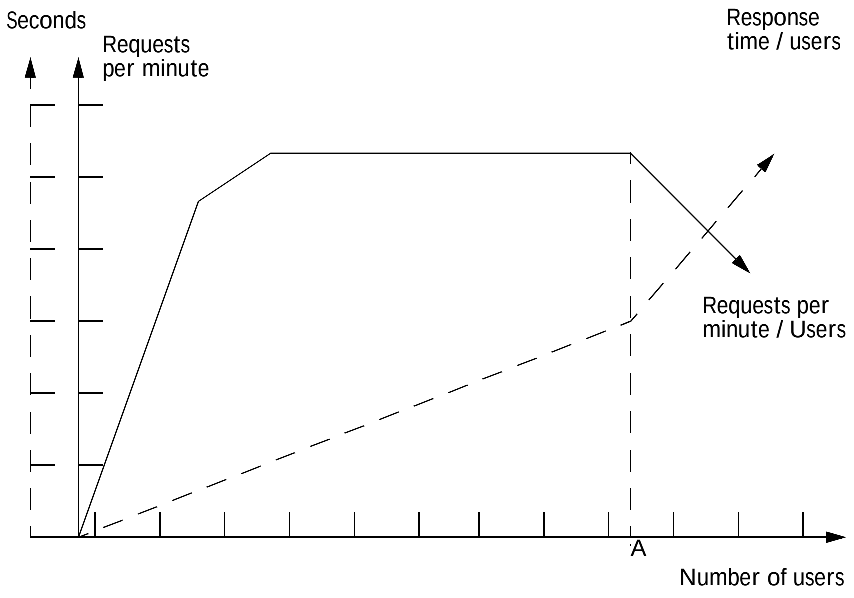
Frequency and Duration of Peak Load

Another issue that may affect whether or not 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 the system's 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 decide that you prefer deploying a system designed around steady state load. This means that 10 percent of the time your system's response time will be slower than the other 90 percent of the time. You 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.

Determining Average Response Time

As explained in Chapter 2, "Planning Your Environment," response time refers to the amount of time it takes for request results to be returned to the user. This figure 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 mean, or average, response time. Each type of request has its own minimal response time, but when evaluating system performance, analyze based 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 due to thrashing, as the following diagram illustrates:



A system performance graph like the one above indicates that at peak capacity, 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).

You should always deploy with peak capacity in mind: determine the maximum number of concurrent users your system *can* support with an acceptable response time. In the above graph, point A represents peak capacity, the point at which requests per minute start to decline. Prior to this, response time calculations are not necessarily accurate because they aren't using 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 the two criteria already discussed in this section: maximum number of users and requests per minute.

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

Response time = (concurrent users / requests per minute) - think time

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

For example, if the following conditions exist:

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

Then, the calculation is as follows:

2 seconds response time = (5,000 users / 1,000 requests per second) - 3 seconds think time

After you have calculated your system's response time, particularly at peak capacity, decide what is an acceptable response time for your enterprise. Response time, along with throughput, is one of the factors critical to NAS performance and improving it 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.

Determining Requests Per Minute

If you know the number of concurrent users at any given time and the response time of their requests and the average user think time at that time, you can determine server capacity, or requests per minute.

Typically, you start by knowing how many concurrent users are on your system. For example, after running some web site performance software, suppose you have calculated that the average number of concurrent users submitting requests on your online banking web site is 3,000. This 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 means you can 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.

Then, because requests per minute and response time become inversely proportional at peak capacity, decide if fewer requests per minute are acceptable as a trade-off for better response time, or alternately, if a slower response time is acceptable as a trade-off for more requests per minute. Essentially, you start playing around with the requests per minute and response time thresholds that you will accept as a starting point for fine-tuning system performance. Then you decide which areas of your system you want to adjust.

The requests per second formula is as follows:

requests per second = concurrent users / (response time + think time)

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

The calculation is as follows:

700 requests per second = 2,800 / (1+3)

700 requests per second translates to 42,000 requests per minute.

Developing an Application Profile

In determining system capacity, you need to consider not only the user load on your system, but the nature and activity of the applications users will run on Netscape Application Server (NAS).

When a user submits a request against your application or applications that run on NAS, one or more application components are invoked, sometimes triggering other components to be invoked. An application component is an element such as a servlet, JSP, EJB, or AppLogic.

This section contains the following topics:

- Complexity of Requests
- Application Partitioning Scenarios

Complexity of Requests

When users submit requests, the requests trigger application activity of a certain nature. The kind of activity that occurs can affect system performance, depending on the level of processing that takes place as a result of the request. Think of requests as fitting into the following general categories:

Complexity of requests	Characteristics
Lightweight	Generates a browse operation that fetches static information, such as an HTML page.
Mediumweight	Generates a search operation consisting of a read-only database query. The information can be cached so as to increase response time the next time the request is generated.
Heavyweight	• Generates a login or logout operation • Generates calculations that write or update information on a database • Creates new records on a database

Examples of different kinds of application requests are described in Chapter 4, “Performance Testing and Fine-Tuning Your System.”

Determine what kinds of application components, and more importantly, what kinds of requests they generate. Perhaps most requests fall into one of the categories listed in the table above, or perhaps you’ll develop a combination. Knowing what mix of requests run on your system helps you to calculate the level of activity that will take place on your server.

Application Partitioning Scenarios

Application partitioning refers to how you deploy and host applications across servers. Application partitioning affects server performance and should be considered when you make decisions about how to improve throughput and response time.

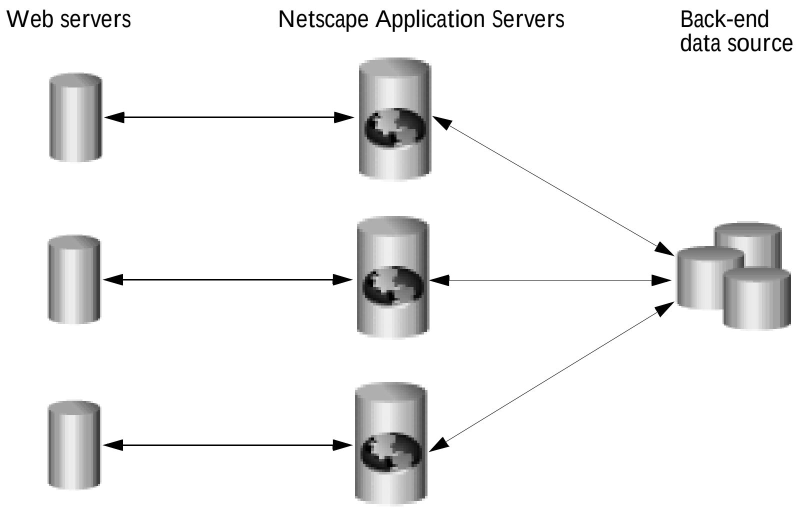
The following table describes three common ways to host an application on multiple NAS machines:

Hosting configuration	Description
Local	The application is installed on each NAS machine and uses multiple web servers to traffic requests to each server. The NAS machines do not communicate with each other.
Partitioned	Parts of the application are hosted on different NAS machines. Each server knows where the application components of the application are hosted on other servers and forwards requests to the appropriate server.
Distributed for load balancing	Parts or all of the application are duplicated on two or more NAS machines. You can then configure the servers to balance application-request loads.

Hosting Applications Locally on Multiple Servers

Hosting applications locally on multiple servers is the simplest of the three most common server configurations. In this configuration, you deploy the complete application on each NAS machine. If the application is already installed on a NAS machine, you can deploy the application to other servers.

Supporting applications locally on multiple servers means that each server stands alone. That is, the two or more NAS machines in the configuration do not communicate with each other. You must have at least one web server for each NAS machine. The following illustration depicts a local hosting configuration:



Local hosting requires that you configure each Web Connector plug-in to forward requests to the appropriate NAS machine.

Hosting Partitioned Applications on Multiple Servers

To partition an application, you must divide up the application components that make up an application. Application components are then hosted by separate NAS machines. Partitioning applications allows each NAS machine to specialize in one type of processing.

For example, servlets responsible primarily for data access are I/O intensive, while servlets responsible for performing calculations are CPU and active-memory intensive. To maximize your application's overall performance, you can partition the application to host these different types of servlets on separate NAS machines. Then you can configure the hardware of the individual servers to support the needs of the application components hosted there.

Keep in mind that if you isolate application components on one machine, you risk losing failover capabilities for that component. For more information about failover planning, see "Developing a Failover Provision," on page 71.

Hosting and Deploying Applications for Load Balancing

Balancing application-request loads, or load balancing, differs from partitioning applications. Load balancing requires that you place one or more copies of an application component on multiple NAS machines rather than simply dividing an application's components among multiple servers (or partitioning the application). You then configure each server, allowing it to find application components on other servers.

When you deploy an application, you must decide if you want to configure the application for load balancing and, if so, how you will configure it. Select the configuration that is most useful for your current scenario; for example, you might have three NAS machines used for testing applications. Your production environment, where users' requests are actually processed, also consists of three NAS machines. Because the application components could be different between the two groups of servers, you do not want to enable application load balancing. Therefore, when you deploy an application from the test servers to the production servers, you should choose only to balance the loads between the destination servers.

Later, should you scale your system to include three more NAS machines in the production group, you can join all the servers in that group when deploying the applications from one of the existing runtime servers to the new servers. The application loads are then balanced between the existing servers and the new servers.

For details about the different application deployment configurations, load balancing, and how to partition applications across servers, see Chapter 12, "Administering Multi-Server Applications," in the *Administration Guide*.

Developing a Failover Provision

Failover, the ability to continue processing server requests without interruption, is one of the general goals of deployment, as explained in Chapter 1, "Overview of Netscape Application Server Deployment." This feature requires that you configure multiple servers (at least two) and processes to provide failure recovery. Netscape Application Server (NAS) distributes and maintains state and session information across servers that are members of the same cluster, allowing requests to be handled by another server in the cluster in case a server becomes unavailable. Thus, there is no single point of failure for a running application.

Planning Failover Capacity

Failover capacity planning means deciding how many additional servers and processes to configure 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 could result, causing response-time degradation or even total loss of service. Preparing for such an occurrence is critical to a successful deployment process. Every enterprise is unique, and failover requirements vary from site to site. Only you can determine what your capacity requirements are based on the kinds of applications you run and hardware resources you have.

However, as a general rule, to ensure failover, make sure of the following:

- You must have redundancy of resource entities: multiple servers, multiple processes, multiple connections between servers and back-end data sources, and so on, such that a backup resource for every entity exists in the event that an outage occurs. This redundancy can be at the application level as well. In a two-machine system, if the first NAS fails, the only way the second NAS can process all requests that would normally be processed by the first NAS is if all applications have been deployed to both machines.
- The remaining entities must have the capacity to continue processing after the outage occurs without getting overloaded.

Determining Your Failover Requirements

Start by determining the number of concurrent users a single NAS machine can support. Then, determine the number of concurrent users the entire system must support at peak load. If a single machine can service 5,000 users and the expected number of concurrent users at peak load is 6,000, running only two machines can certainly handle this volume: NAS distributes the load evenly across the two servers and handles the added overhead of maintaining multiple servers. However, this deployment scenario does not solve failover problems. If one of the two machines fails, the remaining one cannot support the full capacity of 6,000 users at peak. Instead, to plan for failover, you need to deploy three machines. As long as all are fully functional, they each handle 33 percent (2,000 users) of the full peak load, well within the 5,000 concurrent user limit that a single machine can support. Then, in the event that one machine fails, two still remain, across which the full load can be redistributed without overloading the system.

Determining Your Failover Threshold

Assuming that your system requires a total of x number of machines to handle the full peak load *and* provide failover capabilities, when the number of machines that are fully operational equals $x – 1$, you are over your failover threshold, meaning that you do not have enough active servers to handle peak load in the event of a server outage.

Adding servers does not necessarily improve performance linearly. In other words, if your system consists of one NAS machine, adding a second does not mean your system has a twofold improvement in performance. This is because some of the resources of each machine are used to maintain load balancing and state and session management information across the various servers. Adding more servers does improve throughput, but at a cost: with each additional server, the cluster has more maintenance work and failover overhead, and hence the added servers must devote resources to this, preventing the performance from improving linearly.

You must decide two issues:

- What percentage of the total peak load are you comfortable having each machine support?
- Based on this, how many “extra” machines do you want to add to be prepared for server outage circumstances? In other words, how far *within* your failover threshold do you want your system to be?

See Chapter 2, “Planning Your Environment,” for information about determining the appropriate machine and CPU mix for your enterprise.

Adding Server Processes

Besides increasing the number of servers in the cluster to ensure failover, you can add server processes to one or more individual servers. Each NAS server in a cluster is made up of the following four “engines,” or process types:

- Administrative Server (KAS)—An Administrative Server brings up and monitors the other engines and makes sure that any engines that fail are brought up again.
- Executive Server (KXS)—Only an Executive Server can be the primary synchronization engine (the Sync Primary) for a NAS cluster.

  In a cluster of NAS servers, one of the Executive Servers maintains the distributed (synchronized) information and sets up server roles for all the other servers participating in the cluster. All engines in a cluster know how to access this primary engine and the information that is stored on it.
- Zero or more Java Servers (KJS)—A Java Server is the process that hosts servlet, JSP, EJB, and AppLogic objects written in Java.
- Zero or more C++ Servers (KCS)—A C++ Server is the process that hosts AppLogic objects written in C++.

The most basic requirement for successful failover is a backup entity to which requests can be sent. Within a single server, this means that a minimum of two engines must exist for process failover to be successful. You can add a Java Server or C++ Server process to increase fault tolerance. If one process fails, the next one takes over. The failed process is auto-detected by the Administrative Server and automatically restarted using the failure recovery mechanism. You can adjust the number of times the Administrative Server attempts to restart a process, increasing fault tolerance and application availability.

Setting Limits on the Number of Processes

The more processes you run across a cluster of NAS machines, the greater the amount of communication between the machines, causing system resources to be tied up. Furthermore, the more processes you run, the more states that exist at the time of failure.

On a per-CPU basis, follow these guidelines:

Application type	Characteristics
Most common applications	Two processes per CPU
CPU-intensive applications	One process per CPU
I/O-intensive applications	Two to four processes per CPU

Keep in mind that if your system cannot handle additional processes, adding more could negatively affect the performance of an application.

For additional details about determining the number of processes to use, see Chapter 4, “Performance Testing and Fine-Tuning Your System.”

Adjusting the Number of Request Threads

Besides increasing the number of Java Server and C++ Server processes, you can also increase the number of threads each process uses to handle requests. A thread is a system entity owned by a server process, such as a Java Server (KJS) or C++ Server (KCS). A thread handles user requests for applications. NAS assigns requests, or lines of code, from an application to a free thread in a process, so that the thread can manage the resources required by the request. When a resource becomes free, the thread allows the request to use the resource.

Assigning additional threads to a process removes available threads from the systemwide thread pool, limiting the system’s ability to process other thread-utilizing requests, such as those used for database access.

For additional details about determining the number of threads per process, see Chapter 4, “Performance Testing and Fine-Tuning Your System.”

Chapter

4

Performance Testing and Fine-Tuning Your System

This chapter explains how to run performance tests against Netscape Application Server (NAS), analyze the test results, and then, based on the results, how to fine-tune your system to improve performance.

Every application running on NAS is designed to behave and operate differently, based on the business need the application addresses and the enterprise that developed it. There are certain activities and operations, however, that are typical of most applications. This chapter uses an example of an application that exhibits some of these typical behaviors and explains how to run performance tests against this application.

This chapter contains the following sections:

- Goals of Performance Testing
- A Typical Application Test Scenario
- Running Performance Tests
- Analyzing Test Results
- Fine-Tuning Netscape Application Server
- Fine-Tuning Worksheet

Goals of Performance Testing

The main purpose of running performance tests is to determine the performance of your server under specific application conditions. Based on a particular mix of requests and application scenarios, performance testing tells you

- How the server responds to a given realistic load and where the bottlenecks occur before you take the system into production
- How your applications perform
- If your current NAS configuration—made up of web servers, back-end data sources, NAS machines, and a certain number of CPUs—can sustain your peak load with an expected level of performance.

Plan on running tests that represent the application activity on *your* system, using the same hardware as your production environment configuration. You can run nonproduction versions of the actual applications you've already developed. Testing should include loads that represent steady-state activity as well as peak-load activity so that you can determine server behavior and response times under both conditions. The key is to simulate your production environment, activity, and data so as to obtain the most accurate real-world results possible. Make sure to devise an application scenario that includes requests and activity that are typical of your application.

A Typical Application Test Scenario

A typical web-based application generates a mixture of browsing and buying (ordering) requests, such as the model implemented in the Online Bookstore sample application installed with NAS. It also generates other requests, some of which occur more frequently than others.

Request Types

The following table defines the requests that occur in the application testing scenario described in this section:

Request	Description
Browse	Reviews a random group of pages, without involving a Power search. The user randomly goes from one static HTML page to another, by clicking on links. Light operations on the database back-end. Response time should be very fast.
Shopping Cart	Updates a temporary shopping cart record and calculates the total quantity of items in the cart and the total price of the items. Occurs frequently. Databases are not updated. Response time should be fast.
Buy/Place Order	Commits the user's order. Occurs less frequently than a Shopping Cart request. Performs read-write operations to the Order database. Response time should be fast.
Registration	Adds a customer record to the Customer database. Performs a read-write operation. Occurs infrequently, only once per user, compared to other requests such as Buy or Sign In.
Sign in (Log in)	Authenticates the user against the Customer database. Performs a read-only operation on the Customer database. Occurs less frequently than Browse or Shopping Cart requests. Response time should be fast.
Log out	Terminates a user session. Does not affect any database. Response time should be fast.
Search	Goes through the database for a specific item or set of items based on certain search criteria. Performs a read-only database operation. Occurs less frequently than a Shopping Cart request. Does not require as fast a response time as a Shopping Cart or Buy request.

Application Flow

The following scenario uses the NAS Online Bookstore application to simulate a single user's interaction with the application. The scenario can be incorporated into a performance test script to test expected application and server activity.

1. The user enters the URL of the shopping site's home page to navigate to the application.
2. The user pauses 3 seconds (think time) to view the home page and determine which link or button to click.
3. The user clicks Sign In to navigate to the Customer Sign In page.
4. The user pauses 3 seconds (think time) to view the Sign In page.
5. If the user is already registered, she enters her email address and password, and clicks Sign In.
6. If she is not yet registered, she does the following:
    1. Clicks Register Here.
    2. Enters the required data on the Registration Page.
    3. Clicks Save.
7. The user pauses 3 seconds to visually locate the link she will click to search the Book database. (This, plus Step 5—entering email address and password—or Step 6.2—entering the required data on the Registration page—equals think time.)
8. The user clicks Browse Subjects to navigate to the shopping site's list of book subjects.
9. The user pauses 3 seconds (think time) to view the Browse Subjects page.
10. The user clicks a subject name, for example, Literature or Computer, to navigate to the first page of the resulting list.
11. The user pauses 5 seconds (think time) to view and scroll through the page.
12. The user clicks the Add to Cart button next to an item on the list to add it to the Shopping Cart and simultaneously navigate to the Shopping Cart page.

13. The user pauses 3 seconds to view the Shopping Cart page, which displays the total quantity and cost of the items currently in the shopping cart.
14. The user repeats Steps 7 through 13 three times.
15. The user clicks Continue Checkout, which navigates to the Checkout page. The checkout page displays a summary of the committed order.
16. The user pauses 6 seconds (think time) to view the page and ascertain that the ordering information is correct.
17. The user clicks Logout to exit the application.

This scenario simulates a variety of requests with the following frequency:

Request	Frequency	Steps where request is executed
Browse	7	1, 3, 6.1, 8
Search	4	10
Shopping Cart	4	12
Buy/Place Order	1	15
Registration	1 or 0, depending on user	6.3
Sign in (Log in)	1 or 0, depending on user	5
Log out	1	17

In addition to simulating these requests and their frequency, the application scenario also factors in think time, which affects performance calculations, as explained in Chapter 3, “Determining System Capacity.”

Understanding the different requests, when they occur in the application flow, and what parts of the system they affect is important when analyzing test results. If a bottleneck occurs, you can isolate it by removing requests. When you determine which request causes the problem, you’ll know what part of the system to address for improving performance.

Running Performance Tests

Assuming that the scenario described in “Application Flow” on page 80 represents a typical application flow, a performance test using this scenario would involve the following tasks:

1. Prepare your test environment, and make sure it includes the following:
    - Hardware that mirrors your production environment.
    - Firewalls that mirror those in your production environment.
    - A testing tool that predicts and measures system behavior and performance. You want to use a tool that can mimic your enterprise activity by emulating the number of users and requests you expect on your system. After you run the test, such a tool provides statistics and graphs of system behavior to help you analyze performance.
    - The Netscape Application Server Administrator tool. You can use this tool while running tests to measure NAS performance. For information about how to install Netscape Application Server Administrator, see the *Installation Guide*. For information about how to measure NAS performance using the Netscape Application Server Administrator tool, see Chapter 3, “Monitoring Server Activity,” in the *Administration Guide*.

    For performance test results to be accurate, it is very important that your test environment closely match, if not exactly reproduce, your production environment.

2. Decide in advance how many users you expect at peak load and at steady state load.

3. Decide in advance your response time expectations at peak load and at steady state load.

4. Decide in advance your expectations for requests per minute (or per second) at peak load and at steady state loads.

5. Know in advance the durations (start and end times) of peak load and steady state load.

6. Know in advance your user behaviors and the kinds of requests they submit. Your applications determine the mix of requests, based on how they are designed, so know your applications and how your users work with them.

7. Create a script that automates your application scenario.
   - The scenario should map as closely to your user behavior as possible, including think times and the typical mix of requests your users submit.
   - The script should have the ability to simulate multiple users.

8. Run one or both of the following types of tests, depending on the benchmarking or performance testing tool you use and on what you want to measure:

   Peak User Load Test: This test measures how many users your system can handle.
   - Run the script with the number of users you expect at steady state load as a parameter.
   - Make sure to vary the iterations so that in half of them, the user performs a Registration request by registering on the application for the first time, while in others, the user performs a Sign In request by signing on to the application.
   - Increment the number of users over a period of time until you reach the peak load amount. For example, if the initial number of users at steady state is 100, start with that number of simultaneous script iterations and increment this by 20 users every 10 minutes until you reach your peak load of, for example, 300 iterations. Then, continue running the iterations for as long as your peak load typically lasts.

   Free Running Threads Test: This test measures how many requests per minute your system can handle, in other words, it validates the system's throughput.
   - Create an application scenario that does not include think time. It should consist of a variety of raw requests on free-running threads. As soon as a request is submitted, another is submitted immediately following it, and so on, without any think time between requests.

- This test is useful because you don't need multiple machines representing multiple clients and you can run the test in less time than a test that includes think time. The normal throughput restrictions found in the peak user load test are eliminated.

9. Using a web site performance and benchmarking tool, track the following information:
   - Response time of requests at each phase of the test: What is the response time for requests when there's a load of 100 concurrent users? What is the response time when you increase the number to 120 concurrent users? And so on.
   - Requests per minute or per second at each phase of the test.
   - At what point does performance start to degrade? In other words, how many users are on the system concurrently when response time starts to increase and requests per minute start to decrease?

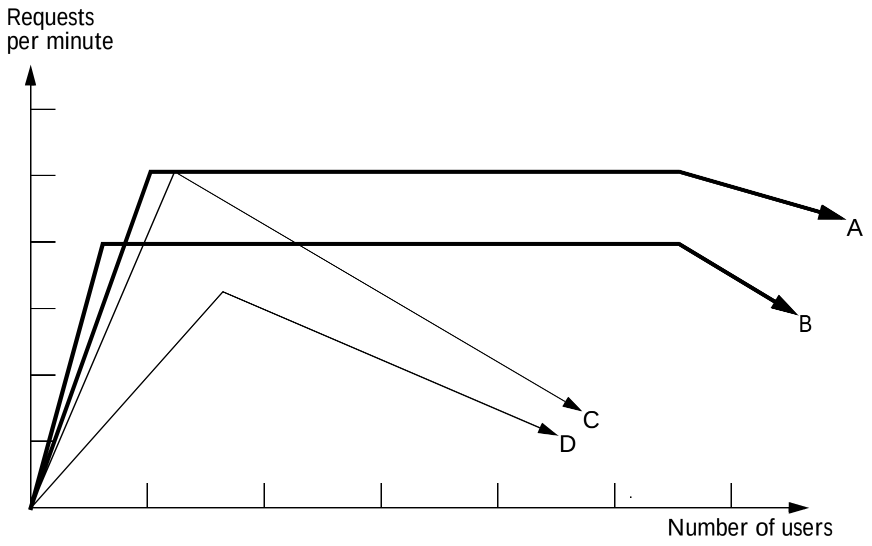
   Tracking this information will help you determine peak capacity information; in other words, information about the maximum number of concurrent users your system can *realistically* sustain, versus what you'd like it to ideally sustain at peak load.

Analyzing Test Results

As explained in Chapter 3, "Determining System Capacity," certain patterns of server behavior are better than others. The benchmarking tools you use to run the performance tests have plotting features for producing graphic representations of test results, or they generate raw data that you can then plot to analyze system performance.

Patterns of System Behavior

The following diagram illustrates the patterns you should look for:



In this diagram, the ideal patterns of server behavior, as already explained in Chapter 3, “Determining System Capacity,” are lines A and B, or any line that maps as closely as possible to A and B. These two lines represent system activity in which CPU resources reach maximum capacity almost immediately (meaning that any number of users, even the initial small number, can take full advantage of CPU resources), a steady state is maintained while the number of concurrent users increases, and then when peak capacity is reached, performance begins to degrade gradually.

Lines C and D represent activity that is less than optimal. In these cases, as soon as peak capacity is reached, performance declines immediately and at a sharp rate.

Based on the patterns of behavior, you then decide if you are satisfied with the results:

- Does the response time for the number of users at a given point in the test cycle meet your response time expectations?

- In particular, does the response time for peak capacity meet your expectations?
- Does the server handle peak load (the *expected* maximum number of concurrent users) without a drop in requests per minute?
- Is the number of concurrent users acceptable when response time starts to increase and requests per minute start to decrease?

Identifying System Bottlenecks

If you are not satisfied with the test results, you must identify where the system bottlenecks are occurring. Examine the various parts of your system to begin isolating the problem.

Analysis Tools

Consider using the analysis tools available with the operating system of the machine where the bottleneck has occurred.

For Solaris machines use the following tools:

- mpstat
- vmstat
- iostat
- netstat
- sar
- proctool

For NT machines use Performance Monitor to look at the resource use of specific components and program processes. This tool creates charts and reports that track the machine's efficiency and identify possible problems.

Possible Bottlenecks

- Is the bottleneck occurring on the client side?

  Although unlikely, sometimes the bottleneck may be in the client that generates requests to the web server. When you run performance tests, always verify if the script or software that simulates user requests has problems at high loads.

- Is the web server performing at near maximum capacity?

  Use the operating system tools on the web server machine to determine if a bottleneck exists in the CPU, memory, disk I/O, network I/O, lock usage, and so on. Additionally, look at the web server administration documentation, or consult with your web server system administrator, for information about how to identify performance bottlenecks.

- Is the back-end data source the problem?

  Use the operating system tools on the back-end data source machine to determine if a bottleneck exists in the CPU, memory, disk I/O, network I/O, lock usage, and so on. For example, if the CPU on the back-end data source is at maximum capacity, while the NAS CPU is, for example, 50 percent idle, then the back-end data source is the bottleneck.

  Additionally, look at the back-end data source administration documentation, or consult with your data source system administrator, for information about how to identify performance bottlenecks.

- Is the application the source of the bottleneck?

  Use an application profiling tool to measure where CPU usage, memory leaks, disk I/O, network I/O, lock usage and other such factors take place in your NAS applications. You may discover that your applications need to be optimized. Refer to the *Programmer's Guide* for information on how to optimize NAS applications.

- Is NAS performing at near maximum capacity?

  Use the operating system tools on the NAS machine to determine if a bottleneck exists in the CPU, memory, disk I/O, network I/O, lock usage, and so on. The Netscape Application Server Administrator tool can help monitor several NAS-related statistics and fine-tune NAS. For more information about monitoring and fine-tuning the server, see Chapter 3, "Monitoring Server Activity," in the *Administration Guide*.

  If you determine that the bottleneck exists on NAS, begin by isolating requests that you ran in the tests and fine-tuning different aspects of NAS, according to the information in the following section, "Fine-Tuning Netscape Application Server."

As a general rule, with regard to all the machines that make up your system (the web server, the back-end data source, and NAS), CPU usage should not be at 100 percent of full capacity, but rather at 90 percent. When a machine's CPU is at full capacity, you have no way of knowing by how much the CPU capacity is exceeded. Furthermore, machines tend to underperform when pushed beyond their capacity, because of the increased contention for the CPU resources.

Fine-Tuning Netscape Application Server

If NAS is the bottleneck in your system, you must fine-tune or tweak your server to improve performance. This section provides fine-tuning guidelines for the following NAS areas:

- Java Server Engines (KJSs)
- C++ Server Engines (KCSs)
- Executive Server (KXS)
- Sessions and Session Data
- Request and Response Objects
- Input and Output of Application Components
- Sticky Application Components
- Load Balancing
- Sync Backups
- Database Connections
- Multiple CPUs
- Cluster Sizing
- Resizing for Throughput
- Web Servers

Java Server Engines (KJSs)

Adjusting the number of KJS engines and the number of threads in each can help improve bottlenecks in NAS. As explained in Chapter 3, “Determining System Capacity,” each NAS server is made up of several engines, or processes. The Java Server engine (KJS) hosts servlet, JSP, EJB, and AppLogic application components written in Java. You specify the number of KJS engines at installation time and can later adjust this number by using Netscape Application Server Administrator.

Adjusting the Number of Engines Per CPU

The number of KJS engines you configure depends on the nature of your application components, particularly servlets, JSPs and AppLogic components. If most of your application components are CPU intensive, in other words, if they perform a lot of computations that expend CPU time and perform very little I/O operations, you should configure one engine per CPU. So, if your NAS machine has four CPUs, configure four KJS engines.

However, if your application components are I/O intensive, configure between two and four engines per CPU, two being the ideal. Even for long-running application components that are not CPU intensive, two KJS engines per CPU is adequate.

To determine the best configuration for your NAS installation, experiment with these numbers under peak load conditions.

Adjusting the Number of KJS Threads

A KJS engine is a multithreaded process, with each thread running an application component, such as a servlet, an EJB, or an AppLogic component. In some instances, depending on how the application has been designed, an EJB takes up multiple threads.

The range of configurable threads in a KJS engine is 16 to 64. You can adjust the number of available threads for each KJS by using Netscape Application Server Administrator. (Or, if you are familiar with the NAS registry, you can reset the Application Server\\4.0\\CCS0 \\ENG\\engine_number\REQ parameter, where *engine_number* represents the engine number of the Java

Server process. This registry parameter sets threads on a per engine basis, overriding the value in the Application Server\\ 4.0\\CCS0\\REQ parameter, which sets the number of threads for all engines.)

Determining the Number of Threads

Typically, for optimal performance, specify between 32 and 48 threads per engine, depending on the request load you expect for your applications. Since a thread can only be used by one application component at a time, when a component is running in it, the thread becomes busy until the component completes running. Therefore, if your applications use many components, you should configure the number of threads accordingly. Specifying more than 48 threads has its drawbacks: a greater amount of memory is required and a greater amount of thread context switching occurs.

For details about specifying KJS engines during installation, see the *Installation Guide*. For details about specifying more KJS engines and adjusting the number of threads, see Chapter 7, “Increasing Fault Tolerance and Server Resources,” in the *Administration Guide*.

C++ Server Engines (KCSs)

The C++ Server engine (KCS) hosts AppLogic application components written in C++. As with KJS engines, you specify the number of KCS engines at installation time, and can later adjust this number by using Netscape Application Server Administrator. KCS engines also support multiple threads.

The recommended number of KCS engines per CPU is the same as with KJS engines and is dependent on the same factors: whether the Applogics are CPU intensive or I/O intensive. A KCS engine can support from 32 to 128 threads, more than a KJS engine can.

For details about specifying KCS engines during installation, see the *Installation Guide*. For details about specifying more KCS engines and adjusting the number of threads, see Chapter 7, “Increasing Fault Tolerance and Server Resources,” in the *Administration Guide*.

Executive Server (KXS)

The Executive Server (KXS) can cause a bottleneck in your NAS configuration. Its role is to accept an application request from the web server, and forward the request to an engine (KJS or KCS) for execution. It also handles the return trip of the request by shuttling the application components back from the KJS and KCS engines to the web server. The KXS does not perform any calculations or heavy-duty operations; it simply controls request traffic. Like KJS and KCS engines, the KXS is a multithreaded process. Each thread handles an application component, such as a servlet, an EJB, or an AppLogic component. When the threads get too busy, throughput is reduced, causing a bottleneck.

Conditions That Affect KXS Threads

The kinds of conditions that make KXS threads busy include:

- Large numbers of requests for small application components that spend very little time in the KXS engine. This condition causes the KXS to work harder and more frequently at redirecting requests.
- Application component result caching. The KXS must update and search the cache to fulfill requests, so when caching is enabled, it affects KXS performance. The amount of data that can be cached is limited by how much physical memory is available. However, caching large amounts of data degrades performance. Typically, the number of cache entries for any instance of an application component shouldn't be more than 100.
- Streaming large amounts of data, such as images and video clips, from the application components. KXS and the Web Connector plug-in performance are affected because all output from an application component must pass through the KXS before reaching the Web Connector plug-in and finally returning to the requesting client.
- Huge request and response streams. When the list of name-value pairs sent to and from an application component (such as a servlet) is large, the KXS becomes busy. For more details about request and response objects, see the section "Request and Response Objects" on page 93.

Determining the Number of Threads

The number of request manager threads in the KXS should be equal to or less than the sum of all the threads in all the KJS and KCS engines that the KXS controls. For example, a KXS that controls four KJS engines, with 32 threads per engine, requires 128 or fewer request manager threads. Fewer than 128 request manager threads is adequate, but adding more will not significantly improve performance. To assist in improving NAS performance, make sure to configure enough threads so that the KXS can always accept a request from a web server without delay. However, configuring more threads than those available in your KJS and KCS engines is unnecessary, since the requests from the KXS will have nowhere to be redirected to until a KJS or KCS thread becomes available.

The maximum number of threads you should configure in the KXS is 256; as a rule, configure 128 request manager threads. To adjust the number of available threads in the KXS, use Netscape Application Server Administrator. (Or, if you are familiar with the NAS registry, you can reset the Application Server\\ 4.0\CCS0\\ENG\\0\\REQ parameter in the NAS registry, where 0 represents the engine number of the KXS process unless the engine number has been manually configured to another value. This parameter overrides the value in the CCS0\\REQ parameter, which sets the number of threads for all engines.)

For details about how to adjust the number of KXS threads, see Chapter 7, “Increasing Fault Tolerance and Server Resources,” in the *Administration Guide*.

Sessions and Session Data

A session is a grouping of information associated with an application user. The size of a session can affect server performance. If you design an application to use sessions (by calling the `getSession()` method for servlets, or the `CreateSession()` method for AppLogics), the information is typically created when the user connects with or logs in to the application. The session is then maintained by NAS until the user exits out of or logs off the application, at which point the session information is destroyed (again, you must explicitly code your application to end the session; this does not happen automatically).

Session Objects

Each session contains a name-value pair list, called the session object, for storing user-specific and interaction-specific information, such as user preferences, security credentials, and shopping cart contents. The name-value pair list is intended to act as a data lookup key, or an index. The session data provides a key that can be used to look up the original data stored elsewhere, such as on a back-end data source. Also, a session is maintained by the Data Synchronization feature of NAS, which provides distributed data synchronization across servers in a cluster.

Limiting the Size of Session Data

Sessions occupy memory and network bandwidth because they must be copied between servers for synchronization purposes. Therefore, keeping sessions small can significantly improve performance when a server participates in data synchronization. Sometimes applications are inadvertently designed in such a way that the session data is treated like a database: the session becomes a channel for pumping large amounts of data to the client. When you design your applications, limit the size of your session data to a few kilobytes, for example 1 to 2 KB, not megabytes. Large numbers of small sessions do not affect performance, but any number of large sessions do.

For information about creating sessions, see Chapter 11, "Creating and Managing User Sessions," in the *Programmer's Guide*.

Request and Response Objects

In addition to the name-value pair list created and maintained by the session, an application component—such as a servlet—has a name-value pair list that is fed into and sent out of it. Every servlet has a name-value pair list, referred to as the request object for application input, and response object for application output. In contrast, a session name-value pair list exists only when you call the `getSession()` method, for servlets, or the `CreateSession()` method, for AppLogics. The servlet name-value pair list contains information such as standard HTTP headers—server name, server post, path information, query strings, and so on—and any cookies that were sent by the requesting client, such as a session ID.

Request Objects

The request object is created as soon as the KXS engine receives a request from the Web Connector plug-in by extracting information from the HTTP request. Typically, the request object contains a list of about 15 to 20 name-value entries. The KXS engine essentially deconstructs and rebuilds the request object before sending it to the servlet. Therefore, the length of the list affects performance. Note that the size of a single entry is not as important; a request object with 10 entries, each containing 1000 bytes of data performs better than a request object with a 100 entries each containing 100 bytes of data.

Guidelines for the Name-Value Pair List

Some guidelines for servlet name-value pair lists will help you improve performance:

- Keep the number of entries in the list to a minimum.

  Even if the name-value pairs in the list are distinct entries, try to bundle them together. NAS does not interpret the data itself when the KXS engine deconstructs the list, so bundle the entries together, using a protocol, and store the bundled entry as a single item in the list. You can then code the application so that it separates the data when it receives the list.

- Keep the number of cookies to a minimum.

  Every cookie becomes a distinct entry in a name-value pair list. Avoid using too many cookies as a way of reducing the number of entries on the list.

For more information about request and response objects, see Chapter 3, “Controlling Applications with Servlets,” in the *Programmer’s Guide*.

Input and Output of Application Components

The amount of data that you send to and from your application can affect server performance. Understanding the flow of data can help you avoid some common bottlenecks that affect system performance.

Packet Flow

As the diagram in the section “Planning Firewall Location” on page 25 illustrates, a request is sent from the client (web browser), to the web server over a TCP connection. After further processing inside the Web Connector plug-in, the request is sent to the KXS engine in NAS, again over a TCP connection. The KXS engine performs further processing of the request and then sends it to the KJS or KCS engine, depending on whether your application components are designed in Java or C++.

Any data that the application sends *back to* the client must first be streamed to the KXS engine, then from there to the Web Connector plug-in, and then from there to the originating web browser client.

Size of Packets

The TCP connection transports the requests as packets of information over network lines, as explained in “Assessing Your Overall Network Configuration” on page 21. Typically, a packet size is about 32 KB, a size that is set by the operating system. Response time will be faster if all the information an application component requires and returns can fit into a single packet. You can also send several packets—for example 256 KB equals 8 packets. However, if you start transferring more than 8—for instance, 10 or more packets—performance will likely degrade as the traffic over the network increases. Given these limitations, try to design your applications so that the size of the HTML page returned to a client in response to a request is no greater than 31 KB. The remaining 1 KB is used by NAS to package HTTP headers, cookies, and so on.

Sticky Application Components

When you develop an application component, you can mark it as sticky. This means that it is processed by the same NAS machine or process where it is initially invoked. For example, a servlet called ShopCart is duplicated on two application servers, Server A and Server B, for load balancing. If ShopCart is invoked on Server B by a particular client, all subsequent sticky requests from that same client for that component are processed on Server B only. In other words, the component “sticks” to Server B. This maintains the integrity of state and session information for an application component that does not distribute session information.

Improving Performance with Sticky Application Components

Sticky application components can improve performance because their session data and other application processing data (such as back-end connections), reside locally in a KJS or KCS engine of a particular NAS machine, for a particular requesting client. In other words, if a request for a sticky application component comes from a client that previously submitted a request for a sticky application component, the request is routed to the same engine that handled the earlier request. If the application component is not sticky, the engine has to retrieve the data from the KXS every time the engine processes the component and the KXS has to determine which KJS or KCS to send the request to.

The improvement in performance that you gain from using sticky application components comes at a cost. Sticky application components cannot be dynamically load balanced, because with load balancing the requests are routed to the least loaded server, no matter what its history is of processing application components. So, with sticky application components, servers cannot share the server processing load. If you don’t use dynamic load balancing, you risk overloading the sticky server with requests. Furthermore, if the sticky server fails for any reason, then none of the sticky application components selected for that server can execute due to the single point of failure. Decide whether or not to mark an application component as sticky according to the particular application and the access pattern of that application’s session data.

For more information about sticky application components, see Chapter 13, “Balancing User-Request Loads,” in the *Administration Guide*.

Load Balancing

Load balancing redirects user requests to the server that is best suited to handle them, either because the server is least loaded or because the application component that processes the requests has historically performed best on that particular machine. Load balancing has a direct impact on server performance and on how quickly user requests are processed. In deciding which server to direct requests to, the load-balancing process takes into account factors such as CPU usage, memory thrashing, hard disk access, and so on. It also factors in individual application component statistics, such as last response time, number of times the application component has been executed on a particular NAS machine, and so on.

Broadcasting and Updating NAS Information

For load balancing to be effective, each server involved in the process must have the most current information about all the other servers. This means that information about the factors that affect load balancing must be broadcast to all the NAS machines, and every NAS machine must monitor and update this information to make load-balancing decisions. Broadcasting information too often, results in a high level of network traffic and could slow down response time. However, if the load-balancing information is not calculated and updated frequently, then application components risk not being optimally load balanced because the information NAS uses to make load-balancing decisions is outdated.

When making decisions about load balancing, you face two major dilemmas:

- How frequently should a NAS server update its load-balancing information?
- How frequently should every NAS installation broadcast its load-balancing information?

Update Interval. A minimum value of 5 seconds and a maximum value of 10 seconds is appropriate in most cases. In general, set the Update Intervals criteria for each server to be twice the response time, under stable conditions, of the most frequently used application component. For example, on a system where the most frequently used application component returns requests in 5 seconds, set the update interval to 10 seconds. Setting it to a more frequent update rate causes the server to do more work and could even alter load-balancing characteristics. Use caution with this calculation; if the response time of a heavily used application component is only 1.5 seconds, do not set the Update Interval to 3 seconds.

Broadcast Interval. As mentioned earlier, broadcasting load-balancing information too frequently will not only increase network traffic, it will also increase the work load of your NAS system as all the servers work to post and gather the information. In general, set the Broadcast Intervals criteria for a server to be twice the value of its Update Interval.

Set the Update Interval and the Broadcast Interval criteria using the Load Balancing tool in Netscape Application Server Administrator.

Monitoring Load-Balancing Information

When you set load-balancing criteria, be patient about the fine-tuning process. Determining the best combination of load balancing criteria takes careful monitoring of your NAS configuration over a period of time, during which you must gather statistics about peak load, your mix of request types, response time averages, bottlenecks, and so on. There is no single load balancing solution for all NAS users, since every system is deployed with different parameters and criteria. As with any aspect of NAS deployment, only you can determine over time the best set of criteria for improving performance of the NAS system deployed at your site.

For more information about load balancing and using Netscape Application Server Administrator to set load-balancing criteria, see Chapter 13, “Balancing User-Request Loads,” in the *Administration Guide*.

Sync Backups

As explained in Chapter 2, “Planning Your Environment,” deploying a Sync Backup in your NAS cluster can affect performance. Every Sync Backup adds overhead, due to all the state and session data from the Sync Primary that is continuously backed up in realtime to the Sync Backup. This leads to a heavier flow of network traffic as well as memory allocation and deallocation, both of which can degrade performance. When your application server is operating at peak load, the Sync Backup has to work even harder because of the amount of information it has to track. As a general rule, use a Sync Backup only if it’s absolutely necessary. For example, if you have a database in which all your server information is stored, you may decide you don’t need a Sync Backup.

Whether or not you decide to use a Sync Backup depends on how you balance the tradeoff between your fault tolerance requirements and your performance requirements.

Database Connections

An application component can establish a database connection to store and access information to and from the database. The number of connections, which you control when you design your application, to and from the application component can affect performance.

Database connectivity issues to keep in mind include:

- The nature of the queries and how long they spend in the database
- The speed of the database
- The size of the data you are retrieving from the database
- The efficiency of the database when handling multiple connections

Setting the Number of Database Connections

Limit the number of database connections to no more than 32. Typically, you can set the number of connections to equal the number of threads per KJS or KCS engine that is processing the particular application component. This guarantees that all the engine threads can communicate with the database concurrently. Note that even if your NAS installation uses more than 32 threads per engine, you should stay within the 32 database connections maximum.

For information about implementing database connectivity in your applications, see the *Programmer's Guide*, Chapter 8, "Handling Transactions with EJBs" and Chapter 9, "Using JDBC for Database Access," and the Sun Java Database Connectivity (JDBC) API Specification. All specifications are accessible from `installdir/nas/docs/index.htm`, where `installdir` is the location in which you installed NAS.

Multiple CPUs

As explained in Chapter 2, "Planning Your Environment," when you plan your NAS topology, you must decide how many machines to include in your cluster and, per machine, how many CPUs to implement. These two factors directly affect performance.

Determining the Number of Engines Per CPU

If you follow these recommendations, note that a typical KXS engine can serve approximately 8 KJS or KCS engines, and 16 KJS or KCS engines on an eight CPU machine. For maximum performance, limit the number of KJS and KCS engines on a four CPU machine to 4 or 6. If you intend to exceed this number of engines, do not use more than 2 engines per CPU.

Memory Considerations

The amount of memory on a NAS machine affects its performance. How much memory you should deploy depends on the number of CPUs on that machine and the nature of your applications. Generally, a four CPU machine performs well with a total of 512 MB of memory. This amounts to 128 MB per CPU. The same machine can perform adequately, though not optimally, with at least 64 MB of memory per CPU. However, if your applications require vast amounts of memory because, for example, they allocate a large number of objects, deploy more, such as 256 MB per CPU on a four CPU machine.

Cluster Sizing

The number of machines in a NAS cluster also affects system performance. Chapter 2, "Planning Your Environment" describes several different topologies. After you run performance tests, revisit that information and decide if a different topology and cluster size might perhaps improve performance.

Resizing for Throughput

Chapter 3, "Determining System Capacity" explains how to plan and measure system performance. After you run performance tests and determine your actual capacity numbers, use the test data to redo the calculations described in Chapter 3, "Determining System Capacity."

Start with the number of concurrent users submitting requests on your system. You can make adjustments to engines and threads per engine based on the number of concurrent requests and the number of CPUs in your system.

Example Sizing Scenario with Lightweight Requests

For example, assume the following scenario:

- You have a single NAS machine system with the following configuration
    - 4 CPUs
    - 4 KJS engines (one KJS per CPU)
    - 32 threads per engine
- On average, 120 requests are submitted concurrently on your system.
- These requests are lightweight (see “Complexity of Requests” on page 68).
- The requests should take no more than 2 seconds to complete.

The 120 requests must run on separate threads, which means that there must be approximately 120 free threads to handle the load. In this scenario, there are indeed approximately 120 free threads (32 threads * 4 KJS engines).

Another way of handling this load would be to deploy two NAS machines, each one running two CPUs with two engines per CPU.

Example Sizing Scenario with More Complex Requests

Now, assume that not all your requests are lightweight. Instead, you have a mix of requests, in which 25 percent are mediumweight and heavyweight. Assume that the processing time for these requests is longer, for example 10 to 15 seconds per request. In this case, 30 threads (25 percent of 120) are always busy at any given time. Therefore, if 120 requests are concurrently submitted, only about 98 requests ((32 * 4)–30) are able to find a free thread to run on. The additional requests must wait for a free thread to become available, slowing down response time.

To improve the response time, you should consider adding resources. One option is to increase the number of threads per existing engine, but this might degrade performance because of too many threads on a few already saturated engines. However, if the requests are primarily of a mediumweight level of complexity (see “Complexity of Requests” on page 68), generating a lot of database queries, rather than a heavyweight level of complexity, generating calculations and new records, performance should be adequate. Again, performance depends greatly on the nature and mix of your system's requests and the applications that respond to those requests.

Overloading Engines

Adding too many engines, or too many threads per engine, to fine tune performance is a common situation that you should try to avoid. Often, this causes performance to degrade, because as mentioned above, the CPUs are already saturated (running at 90 to 95 percent of full capacity) and cannot handle the extra load even with the additional resources. The overhead of thread context-switching adds costs that far outweigh the benefits of additional threads. The additional engines and threads don't even have a chance to run, because the CPU is already busy. The only workaround is to add more CPUs or, if possible, deploy more NAS machines on your system.

Anticipating Future Load

Add extra resources, if you can, beyond those you know you'll need to meet peak load requirements. Even if you do not initially add more engines to improve throughput, plan your hardware for the possibility that you will add them in the future as throughput on your server increases, which often happens over time. In other words, deploy multiple CPUs up front in anticipation that you may need to add more engines later. For example, in the scenario described in this section, "Example Sizing Scenario with Lightweight Requests," you may decide to deploy two machines with four CPUs each but only run two engines per machine. This leaves you adequate room to add more engines later, if and when you need them.

Web Servers

As explained in Chapter 2, "Planning Your Environment," the number of web servers in your NAS topology can affect performance. After running performance tests, determine if the web server is causing the bottleneck in your overall system.

The web server's role is to accept client requests, package them, send them to the KXS engine in your NAS system, receive the response packets from NAS, and deliver them back to the originating client. The Web Connector plug-in assembles headers (HTTP headers, query strings, cookies, and so on) and sends the packets to the KXS.

Setting the Number of Web Server Threads

Netscape Enterprise Server (NES) can accept approximately 512 concurrent client requests. However, because NES has a smaller back-end thread pool (referred to as the worker threads) to run requests on the Web Connector plug-in, not all of these requests can be concurrently forwarded on to the KXS engine in NAS. You should configure the number of worker threads to be as low as possible, so as to sustain the optimal volume of requests. A high number of threads may allow more requests to be processed concurrently but results in more CPU overhead due to thread context switching.

If your web server is installed on a separate machine from your NAS system, and the CPUs on the web server machine are not operating at full capacity but you notice client requests and responses being queued up, you should increase the number of worker threads. The limit on the number of worker threads is set by the NSCP_THREADPOOL_MAX environment variable in NES. The default value, 100, is usually adequate.

If the CPUs on the web server machine are operating at full capacity and those on the NAS machines are not, then consider adding additional web server machines. Usually, the CPUs on the NAS machine become fully saturated first, but if your applications are returning large amounts of data and a lot of processing is taking place in the Web Connector plug-in, then the Web Connector plug-in could be the bottleneck.

For details about fine-tuning your web server, refer to your web server documentation and consult your web server administrator.

Fine-Tuning Worksheet

The following worksheet summarizes information in the previous section, and helps you track decisions you've made about your NAS configuration. Read the section "Fine-Tuning Netscape Application Server" on page 88 for details about each area of fine-tuning.

This worksheet does not substitute the material covered throughout this *Deployment Guide*. You should read this guide carefully and use all the information presented when making your deployment decisions.

General System Area Client that is generating requests

Bottleneck When running performance tests, the script or tool that generates the requests may have problems with high loads.

Recommendation Check the script

Your Deployment Decision/Action

General System Area Web server

Bottleneck CPU, memory, disk I/O, network I/O, lock usage, and so on

Recommendation

- Use operating system tools on the web server machine to verify the specific location of the bottleneck on that machine.
- Deploy more web servers.
- Use a web server load balancer, such as Cisco's LocalDirector to optimally distribute requests across all your to the web servers.
- Configure the number of worker threads to be as low as possible.
- See the web server administration documentation.
- Consult with your web server administrator.

Your Deployment Decision/Action

General System Area Back-end data source

Bottleneck CPU, memory, disk I/O, network I/O, lock usage, and so on

Recommendation

- Use operating system tools on the back-end data source machine to verify the specific location of the bottleneck on that machine.
- If your back-end data source is a database, limit the number of database connections to no more than 32. For a configuration of less than 32 threads, set the number of connections from the application to equal the number of threads per KJS or KCS engine that is processing the particular application component.
- For information about implementing database connectivity in your applications, see the
  - *Programmer's Guide*, Chapter 8, "Handling Transactions with EJBs"
  - *Programmer's* Guide, Chapter 9, "Using JDBC for Database Access"
  - Sun Java Database Connectivity (JDBC) API Specification.All specifications are accessible from `*installdir*/nas/docs/index.htm`, where *`installdir`* is the location in which you installed NAS.
- See the back-end data source administration documentation.
- Consult with your data source administrator.

Your Deployment Decision/Action

General System Area NAS

Bottleneck CPU, memory, disk I/O, network I/O, lock usage, and so on

Recommendation

- Use operating system tools on the back-end data source machine to verify the specific location of the bottleneck on that machine.
- Use Netscape Application Server Administrator to monitor and fine-tune the server. For more information about this, see Chapter 3, "Monitoring Server Activity," in the *Administration Guide.*

Your Deployment Decision/Action

General System Area NAS

Bottleneck Executive Server (KXS)

Recommendation

- Avoid developing applications that generate large numbers of requests for small application components that spend very little time in the KXS engine.
- Avoid caching large amounts of data. Keep the number of cache entries in any application component at no greater than 100.
- Avoid streaming large amounts of data, such as images and video clips, from your application components.
- Avoid creating huge name-value pair lists in the request and response objects that get sent to and from your application components.
- Configure no more than 128 threads. If you configure less than this number, make sure the number of KXS request manager threads is equal to or less than the sum of all threads in all the KJS and KCS engines that the KXS controls. For details, see Chapter 7, "Increasing Fault Tolerance and Server Resources," in the *Administration Guide.*

Your Deployment Decision/Action

General System Area NAS

Bottleneck Java Server Engines (KJSs)

Recommendation

- Adjust the number of KJS engines per CPU:
  - A typical KXS can serve approximately 8 KJS engines.
  - For maximum performance, limit the number of KJS engines on a four CPU machine to 4 or 6. If you exceed this number, do not use more than 2 engines per CPU.

  - If your application components are CPU intensive, configure one engine per CPU.
  - If your application components are I/O intensive, configure between two and four engines per CPU, two being the ideal.
- Adjust the number of threads per KJS:
  - If your application uses many components, configure the number of threads accordingly.
  - Try not to specify more than 48 threads per engine.
- For details about specifying more KJS engines and adjusting the number of threads, see Chapter 7, "Increasing Fault Tolerance and Server Resources," in the *Administration Guide*.

Your Deployment Decision/Action

General System Area NAS

Bottleneck C++ Server Engines (KCSs)

Recommendation

- Adjust the number of engines per CPU:
  - A typical KXS can serve approximately 8 KCS engines.
  - For maximum performance, limit the number of KCS engines on a four CPU machine to 4 or 6. If you exceed this number, do not use more than 2 engines per CPU.
  - If your application components are CPU intensive, configure one engine per CPU.
  - If your application components are I/O intensive, configure between two and four engines per CPU, two being the ideal.
- Adjust the number of threads per KCS engine (between 32 and 128 threads).
- For details about specifying more KCS engines and adjusting the number of threads, see Chapter 7, "Increasing Fault Tolerance and Server Resources," in the *Administration Guide*.

Your Deployment Decision/Action

General System Area NAS

Bottleneck Load Balancing

Recommendation

- Set the Update Intervals criteria for each server to be twice the response time, under stable conditions, of the most frequently used application component. A minimum value of 5 seconds and a maximum value of 10 seconds is appropriate in most cases.
- Set the Broadcast Intervals criteria for a server to be twice the value of its Update Interval.
- For more information about load balancing and using Netscape Application Server Administrator to set load-balancing criteria, see Chapter 13, “Balancing User-Request Loads,” in the *Administration Guide.*

Your Deployment Decision/Action

General System Area NAS

Bottleneck Cluster Size

Recommendation

- Evaluate the number of machines in your cluster and decide if you should add more.
- See Chapter 2, “Planning Your Environment” for more information.

Your Deployment Decision/Action

General System Area	NAS
Bottleneck	Sync Backup
Recommendation	• Use only if absolutely necessary. • For more information, see “Resizing for Throughput” on page 100, and Chapter 2, “Planning Your Environment.”
Your Deployment Decision/Action	

General System Area	NAS
Bottleneck	Memory
Recommendation	• Decide how much memory to deploy based on the number of CPUs in your NAS machine. • For example, a four CPU machine performs well with a total of 512 MB of memory, which amounts to 128 MB per CPU. • If your applications require more memory, deploy more, such as 256 MB per CPU on a four CPU machine.
Your Deployment Decision/Action	

General System Area	Application
Bottleneck	CPU, memory, disk I/O, network I/O, lock usage, and so on
Recommendation	• Use an application profiling tool to measure where CPU usage, memory leaks, disk I/O, network I/O, lock usage and other such factors take place in your NAS applications. • Refer to the *Programmer's Guide* for information on how to optimize NAS applications
Your Deployment Decision/Action	

General System Area	Application
Bottleneck	Sessions and Session Data
Recommendation	• When using data synchronization, keep sessions small. They should be no greater than 1 to 2 KB in size. • For information about creating sessions, see Chapter 11, “Creating and Managing User Sessions,” in the *Programmer's Guide*.
Your Deployment Decision/Action	

General System Area	Application
Bottleneck	Request and Response Objects
Recommendation	• Keep the number of entries in the request and response objects to a minimum. • Keep the number of cookies to a minimum. • For more information about request and response objects, see Chapter 3, “Controlling Applications with Servlets,” in the *Programmer’s Guide*.
Your Deployment Decision/Action	

General System Area	Application
Bottleneck	Packet Flow
Recommendation	• Improve response time by making sure that all the information an application component requires and returns can fit into a single packet. • Instead of fitting all the information into one packet, send several packets, but not exceed 8 packets. • Limit packet sizes to 31 KB.
Your Deployment Decision/Action	

General System Area	Application
Bottleneck	Sticky Application Components
Recommendation	• Decide whether or not to mark application components as sticky to improve performance. • For details, see Chapter 13, “Balancing User-Request Loads,” in the *Administration Guide.*
Your Deployment Decision/Action	

Chapter

5

Deploying the Production System

After you have planned your environment, determined your system capacity, and run performance tests, it's time to deploy your production system and take Netscape Application Server (NAS) to a live staging environment.

This chapter discusses guidelines for bringing NAS into production, and includes the following sections:

- Deploying Applications
- Deploying a Pilot System
- Deployment Checklist
- Fine-Tuning the Production System

Deploying Applications

After you install Netscape Application Server (NAS), you must deploy your applications. Typically, you develop an application on a development machine. Before you can use it, however, you must deploy the application to a destination server, in other words, one of the NAS machines that you have deployed in your enterprise.

For information about deploying and upgrading applications, see Chapter 2, “Deploying and Upgrading Applications,” in the *Administration Guide*.

Deploying a Pilot System

At this point in the deployment cycle you should have a good idea what your system environment is, what kind of capacity it can support, what your system and application needs are, and what steps you can take to improve system performance. However, before you take NAS live into your enterprise, you might want to consider running a pilot program first. The advantages of having a pilot phase are that you can:

- Use the pilot phase to train employees on the new system.
- Incorporate and fine-tune existing business procedures and practices into the new system.
- If you are upgrading from an earlier release of NAS, you can migrate your system data to the new version and eliminate any issues that might arise before you go live.
- Identify real bottlenecks in a live environment without slowing down your current enterprise day-to-day operations.
- Identify real bottlenecks based on real activity against your system. You may have run tests in Chapter 4, “Performance Testing and Fine-Tuning Your System” that were very close to your live environment; however, no matter how closely these tests mirrored system activity, they were run in a controlled situation without real customers. A pilot phase gives you the most accurate data you can obtain, short of going into full production mode.

Deployment Checklist

The following checklist runs through information to gather and think about before you take Netscape Application Server (NAS) into production. These questions are explored throughout this guide and in other NAS documentation.

- Who is your system administrator, and does he or she understand NAS administration? *(Administration Guide)*
- Have you designed and developed your NAS applications? *(Programmer's Guide)*
- What are your capacity goals? *(Chapter 1, Chapter 2, and Chapter 3)*
  - — How many concurrent users do you need to support?
  - — What are your response time goals?
  - — What are your goals for the number of requests per minute, second, or both?
  - — What is the average think time on your system?
  - — How many requests per second/minute do you want your system to support?
- How many web servers will you use? *(Chapter 2)*
- Will the web servers be on the same machine as NAS? Or will they be on separate machines? *(Chapter 2)*
- Do you intend to set up firewalls? *(Chapter 2)*
  - If so, where will the firewalls be located?
  - How will the firewall filters be configured to handle connections, requests, and replies?
- How many clusters will you deploy? *(Chapter 2, Chapter 4)*
- How many servers in each cluster? *(Chapter 2, Chapter 4)*
- Will you deploy a Sync Backup *(Chapter 2, Chapter 4)*
- If you enable data synchronization, how will you set up the cluster? Will you have a failover recovery plan? *(Chapter 2, 3, 4)*
- What does your NAS topology look like? *(Chapter 2)*
- What is your machine and CPU mix? *(Chapter 2)*
- How do you intend to integrate back-end data sources? *(Chapter 2)*

- Will you use global transactions? *(Chapter 2)*
- How are you configuring NAS with Directory Server? *(Chapter 2)*
- What is the complexity of the requests generated by your applications? *(Chapter 3, Chapter 4)*
- How do you intend to handle application partitioning and load balancing? *(Chapter 3, Chapter 4)*
- What is your failover provision? *(Chapter 3, Chapter 4)*
- Have you developed an application scenario for testing and have you automated it with a script? *(Chapter 4)*
- After running performance tests, have you checked the following parts of your system to identify possible bottlenecks? *(Chapter 4)*
    - Client
    - Web servers
    - Applications
    - NAS
    - Back-end data sources

Fine-Tuning the Production System

After you install and start to use Netscape Application Server, you will likely find some bottlenecks that you thought you had resolved during the stress and performance testing phase. Take the opportunity during the pilot system phase, and after deploying the final production system, to continue fine-tuning your system following the suggestions made in Chapter 4, “Performance Testing and Fine-Tuning Your System.”

Glossary

application component	A programmable element of an application. Examples of application components are servlets, JSPs, EJBs, and AppLogics.
cluster	A group of NAS installations that participate in distributed synchronization of state and session data. A cluster is not a group of NAS servers that simply load-balance incoming requests among themselves. The NAS use of this term refers specifically to data synchronization, which servers do not participate in by default. You set up a cluster by assigning a role to each server in the cluster, either when you install the server or later by adjusting settings in the registries of all the cluster members.
concurrent users	Users who are on the system simultaneously, whether they are actively submitting requests or merely in think time mode (viewing data, as opposed to processing requests). Even if a user is not submitting a request, he or she is still adding load to system resources by being actively logged on the system.
deployment	The process of setting up NAS at your site. In this book, deployment refers not to deployment of applications but to deployment of the entire server system.
firewall topology	A schematic layout of where the firewall exists within your network and how the rest of your enterprise interacts with it. See also **topology**.
load balancing	The process by which user requests are sent to the server with the least load, and away from servers too busy to handle additional requests.
name-value pair list (application)	See **request/response objects**.
name-value pair list (session)	See **session object**.
peak capacity	The maximum number of concurrent users that the system can sustain before requests per minute start to decline and response time starts to increase.
peak load	The maximum number of concurrent users that the system should ideally support based on the pattern of activity that typically exists on your system.
performance	Performance consists of two measurable factors: primarily throughput and, in some instances, response time.

request	A single user's request for data, and the return of that data by the server. The request makes a round trip: from the user submitting the request, to the server, and then back from the server that is returning the result of that request to the user.
requests per minute	The number of round-trip transactions that the system can handle per minute. Used as a measurement of NAS performance.
request/response objects	The input and output list sent to and from an application component, such as a servlet. Contains information such as standard HTTP headers—including server name, server post, path information, query strings, and so on—and any cookies that were sent by the requesting client such as a session ID. This list is created automatically as part of your application processing.
response time	The speed at which a request is processed. The faster the response time, the more requests per minute being processed. Unit = seconds (or minutes).
scalability	The ability to support an increase in users and requests, also referred to as an increase in load, by proportionally increasing hardware resources, without degrading response time or system stability. In other words, the system's ability to cope with an increase in users and requests when that increase is matched with a proportional increase in hardware resources. If response time degrades, despite the addition of hardware resources in proportion to the increase in concurrent users, then the system is not scalable.
scaling factor	The overhead that results from adding more servers to a cluster. Adding servers improves throughput, but each additional server introduces more overhead in the form of load balancing and data synchronization. The added servers must devote resources to the scaling factor as well as to the goal of improving performance.
session	A grouping of information associated with an application user. The information is created only if you explicitly call the `getSession()` method, for servlets, or the `CreateSession()` method, for AppLogics, in your application design.
session object	A list of name and value pair entries of data associated with a session. The name and value pair entries are user-specific and interaction-specific information, such as user preferences, security credentials, shopping cart contents, and so on.

steady state	The median level of traffic on your NAS system or over your network. When your NAS system reaches steady state, it maintains capacity, or a steady number of requests per minute, while the number of concurrent users continues to increase. In steady-state mode, your system is not processing at peak capacity; however, it is maintaining a steady capacity and continues to maintain this capacity, even as the number of concurrent users on the system rises.
sticky application component	An application component that has been intentionally marked to be processed by the same NAS machine or process where it was initially invoked.
think time	The time a user spends reviewing and analyzing the result of a processed request before submitting a new request. The user is not performing any actions against the server, and thus a request is not being generated during think time.
throughput	Capacity, or the number of requests that NAS can service in a given time period. A request consists of a single user's attempt to access data, and the return of that data by the server. Throughput is measured in requests per minute or requests per second.
topology	A schematic layout of your network. It is the logical map of your servers, hosts, clients, and other elements, and it shows the connections made between them.

Index