

Best Practices for High-Volume Web Sites



Designing for scalability



Planning for growth



Managing Web sites for
performance



Authored by the
The High-Volume Web Sites Team



International Technical Support Organization

Best Practices for High-Volume Web Sites

December 2002

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (December 2002)

This edition includes papers published by IBM's High-Volume Web Sites Team.

© Copyright International Business Machines Corporation 2002. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	v
Trademarks	vi
Preface	vii
The team that wrote this redbook	viii
Become a published author	xii
Comments welcome	xii
Part 1. Best practices	1
Chapter 1. Knowing your workload	3
Introduction to workload patterns	6
Understanding your workload	7
Chapter 2. Designing for scalability	11
Introducing scalability	12
Six steps to scaling your infrastructure	14
Additional techniques	19
Common pitfalls	21
Choosing between two and three tiers	22
Emerging standards and technologies	23
Summary	25
Chapter 3. Designing pages for performance	27
Introducing Web communications	28
When bad things happen to good pages	30
What's a good page?	33
Design practices that can improve performance	34
It is not just about satisfying customers	37
Chapter 4. Planning for growth	39
Introducing a methodology for capacity planning	40
Summary	56
References	57
Chapter 5. Maximizing Web site availability	59
Introduction	60
Availability concepts and costs	60
On the way to continuous availability	64
Common inhibitors	64
Common techniques	65
Summary	74
References	75
Part 2. Customer engagements	77
Chapter 6. Managing Web site performance	79
Step 1. Establish performance objectives	81
Step 2. Monitor and measure the site	82
Step 3. Analyze and tune components	83

Step 4. Predict and plan for the future	85
Some performance management scenarios	87
Tools for monitoring performance	91
Summary	95
References.	96
 Chapter 7. Charles Schwab puts growth plan to the test	 97
Schwab's e-business today	97
Making sure the new architecture measures up	98
IBM benchmark	99
Schwab's response to the benchmark results	100
Technical benchmark details	101
 Chapter 8. Fine-tuning the scalability of a multi-tier architecture	 105
Introducing the Barista project	106
Testing Barista	107
Summary of Barista results	113
Best practices.	114
Summary	116
Software and test tools.	118
Monitoring tools	118
 Chapter 9. Improving the scalability of a WebSphere application with multihome servlets	 119
Overview of scaling servlets.	120
Multihome servlets	122
Benefits of multihome servlets	123
Summary	127
References.	127
 Related publications	 129
Referenced Web sites	129
How to get IBM Redbooks	130
IBM Redbooks collections.	130
 Index	 131

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM eServer™
Redbooks(logo)™ 
IBM®
iSeries™
MQSeries®
POWERparallel®
pSeries™

PTX®
Redbooks™
RS/6000®
SP™
ThinkPad®
Tivoli®
Tivoli Enterprise™

Tivoli Enterprise Console®
WebSphere®
z/OS™
zSeries™
DB2®

The following terms are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both:

Lotus®

Word Pro®

The following terms are trademarks of other companies:

Sun Solaris is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

LoadRunner is a trademark of the Mercury Interactive Corporation in the United States, other countries, or both.

Oracle is a trademark of the Oracle Corporation in the United States, other countries, or both.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Preface

For more than three years, IBM's High-Volume Web Sites (HVWS) team has been working with many of the world's largest Web sites. The team has accumulated a significant amount of knowledge and defined best practices for designing and deploying high-volume sites, earning a reputation as one of the world's leading centers of expertise on scalable e-business infrastructures. The team has locations in California, New York, Japan, Korea, China, Taiwan, and the United Kingdom.

The IT infrastructures that comprise most high-volume sites present unique challenges in design, implementation, and management. While actual implementations vary, Figure 0-1 shows a typical e-business infrastructure comprised of several tiers. Each tier handles a particular set of functions, such as serving content (Web servers, such as the IBM HTTP Server), providing integration business logic (Web application servers, such as the WebSphere Application Server), or processing database transactions (transaction and database servers). Site workloads are assumed to be high volume, serving dynamic, volatile data.

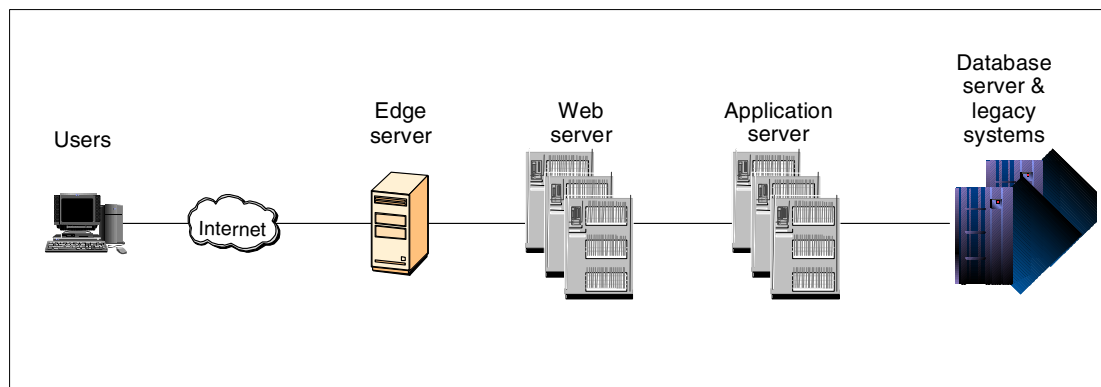


Figure 0-1 Multi-tier infrastructure for e-business

The HVWS team analyzes site traffic patterns to improve performance and availability. Figure 0-2 shows how IBM's HVWS team defines the life cycle of a Web site; it shows also the categories of best practices recommended for one or more phases of the cycle.

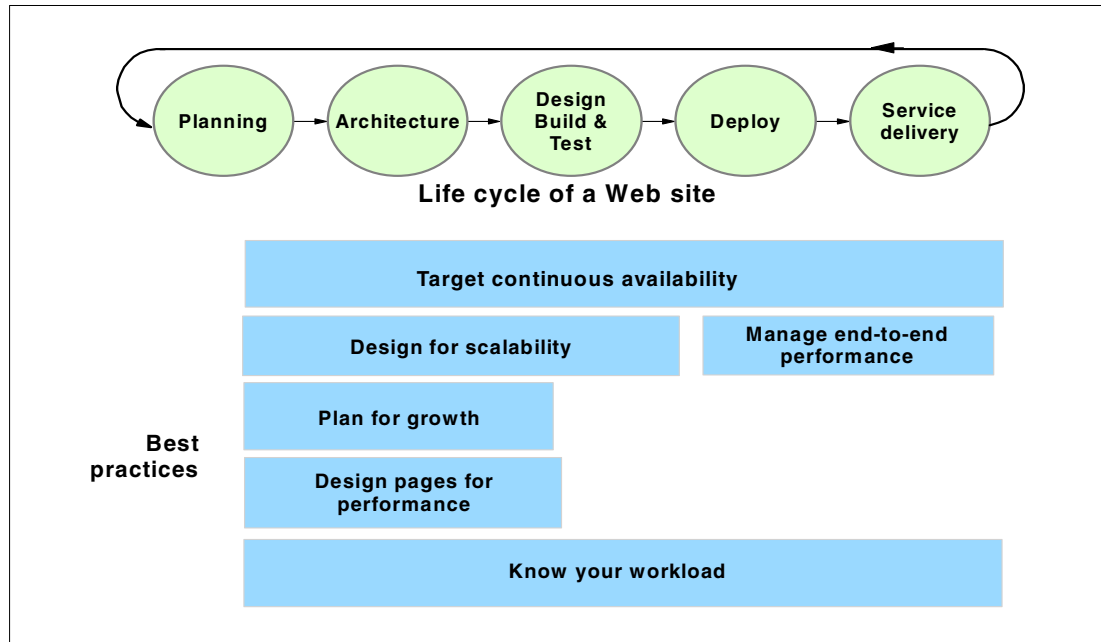


Figure 0-2 Life cycle of a Web site

As it accumulates experience and knowledge, the HVWS team publishes papers aimed at helping CIOs and others like you understand and meet the new challenges presented during one or more of the phases. This IBM Redbook is a compilation of the HVWS papers, which are available individually at the HVWS Web page.

The team that wrote this redbook

Many IBMers contributed effort and information to these chapters and projects.

This first edition contains nine chapters divided into two parts. Part 1 has chapters dedicated to the HVWS best practices. Part 2 contains chapters describing some customer engagements that characterize the kind of work we do with our customers.

Part 1 contains our best practices information.

Knowing your workload is the foundation of our recommended best practices for high-volume Web sites. Chapter 1, **Know your workload**, was written by:

- ▶ Willy Chiu
- ▶ Paul Dantzig
- ▶ Harish Grama
- ▶ Rich Grega
- ▶ Linda Legregni
- ▶ Joe Spano

with contributions from the Super Scalable Architecture team:

- ▶ Maggie Archibald
- ▶ Michael Conner
- ▶ Daniel Dias

- ▶ Greg Flurry
- ▶ Parag Gondhalekar
- ▶ Leonard Hand
- ▶ Richard McDonald
- ▶ Mark Palmer

This chapter introduces common workload patterns and provides the information you need to characterize your workloads.

Optimizing for scalability remains a significant challenge for e-businesses as they balance the demands for availability, reliability, security, and high performance. Vendors are responding with infrastructure options and supporting hardware and software platforms that address these requirements. Chapter 2, **Design for scalability**, was written by:

- ▶ Willy Chiu
- ▶ Paul Dantzig
- ▶ Harish Grama
- ▶ Rich Grega
- ▶ Linda Legregni
- ▶ Joe Spano

with contributions from the Super Scalable Architecture team:

- ▶ Maggie Archibald
- ▶ Michael Conner
- ▶ Daniel Dias
- ▶ Greg Flurry
- ▶ Parag Gondhalekar
- ▶ Leonard Hand
- ▶ Richard McDonald
- ▶ Mark Palmer

This chapter identifies current products and emerging trends that are most likely to improve the scalability of your e-business infrastructure.

Web page performance can make or break a Web site. Chapter 3, **Design pages for performance**, was written by:

- ▶ Mike Amerson
- ▶ Gerry Fisher
- ▶ Larry Hsiung
- ▶ LeRoy Krueger
- ▶ Nat Mills

This chapter discusses the significance of the time it takes to download a Web page and introduces page design practices that can reduce download time and improve resource utilization.

As e-business and its related requirements grow at "Web speed", a critical issue is whether the IT infrastructure supporting the Web sites has what it needs to provide available, scalable, fast, and efficient access to the company's information, products, and services. Chapter 4, **Plan for growth**, was written by:

- ▶ Jerry Cuomo
- ▶ Alan Emery
- ▶ Larry Hsiung
- ▶ Mike Ignatowski
- ▶ Zhen Liu
- ▶ Mark Squillante
- ▶ Noshir Wadia
- ▶ Cathy Xia
- ▶ Li Zhang

This chapter discusses a methodology for modeling your high-volume Web site so that proposed changes can be analyzed to determine how performance objectives can best be met.

High availability is more important than ever now that your customers, suppliers, and/or employees rely on your Web site. Chapter 5, **Maximize Web site availability**, was written by:

- ▶ Tom Alcott
- ▶ Scott Bryant
- ▶ Mike Fitzgerald
- ▶ Ebbe Jalser
- ▶ Tricia Jiang
- ▶ Bob Kalka
- ▶ Richard McDonald
- ▶ Patrick McMahon
- ▶ Scott Sims

This chapter reviews availability concepts and practices that can help you achieve your availability objectives. It includes a summary of practices that pertain specifically to e-business infrastructures.

As enterprises implement Web applications in response to the pressures of e-business, managing performance becomes increasingly critical. Chapter 6, **Manage Web site performance**, was written by:

- ▶ Willy Chiu
- ▶ Jerry Cuomo
- ▶ Ebbe Jalser
- ▶ Rahul Jain
- ▶ Frank Jones
- ▶ W. Nathaniel Mills III
- ▶ Bill Scully
- ▶ Joe Spano

- ▶ Brad Willard
- ▶ Ruth Willenborg
- ▶ Helen Wu

This chapter introduces a methodology for managing performance from one end of the e-business infrastructure to the other. It identifies some best practices and tools that help implement the methodology.

Part 2 has three chapters and describes some customer engagements that characterize the kind of work we do with our customers.

Chapter 7, **Charles Schwab puts growth plan to the test**, was written by:

- ▶ Willy Chiu
- ▶ Keith Jones

This chapter describes a joint project between IBM and Charles Schwab and Co., Inc. to develop an architecture for Schwab's Web site that could cope with soaring growth. It includes a description of the configuration, the test itself, and the test results.

Chapter 8, **Fine-tuning the scalability of a multi-tier architecture**, was written by:

- ▶ Lisa Case-Hook
- ▶ Mike Crumbliss
- ▶ Paul Edlund
- ▶ Harold Hall
- ▶ Bob Maher
- ▶ Luis Ostdiek
- ▶ Joe Spano
- ▶ Ted Tran
- ▶ Jay Warfield

describes the later phases of the project introduced in Chapter 7. During these phases, the project team separated the business logic and presentation layers into different tiers and tested various scenarios to find which topology provides the best performance.

Chapter 9, **Improving the scalability of a WebSphere application with multihome servlets**, was written by:

- ▶ Ranjit Nayak
- ▶ Ursula Richter
- ▶ Ted Tran
- ▶ Fred Tucci
- ▶ Jay Warfield
- ▶ Helen Wu

This chapter describes an engagement in which the HVWS team assessed the performance and scalability of an online trading application in a large commercial bank and found a major bottleneck in application memory. They developed the multihome servlet technique as an alternative to servlet cloning.

Thanks to the following people for their contributions to this Redbook project:

Susan Holic, IBM HVWS Team, Silicon Valley Laboratory, San Jose, California

Linda Legregni, Manager, HVWS Team Project Office, Silicon Valley Laboratory, San Jose, California

Joe DeCarlo, Manager, International Technical Support organization, San Jose, California

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- Send your comments in an Internet note to:

redbook@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. 1WLB Building 80-E2
650 Harry Road
San Jose, California 95120-6099



Part 1

Best practices

Part 1 is a compilation of the High-Volume Web Sites team best practices.

Knowing your workload

As Figure 0-2 on page viii shows, knowing your workload is the foundation of our recommended best practices for high-volume Web sites. It is key during all phases of the life cycle.

Most high-volume Web sites experience volumes that can vary widely on a seasonal or other cyclical basis, or that exhibit burstiness as a result of sudden and unpredictable changes in user demand. Figure 1-1 shows how the volumes of four different actual Web sites can vary by as much as a factor of five to ten.

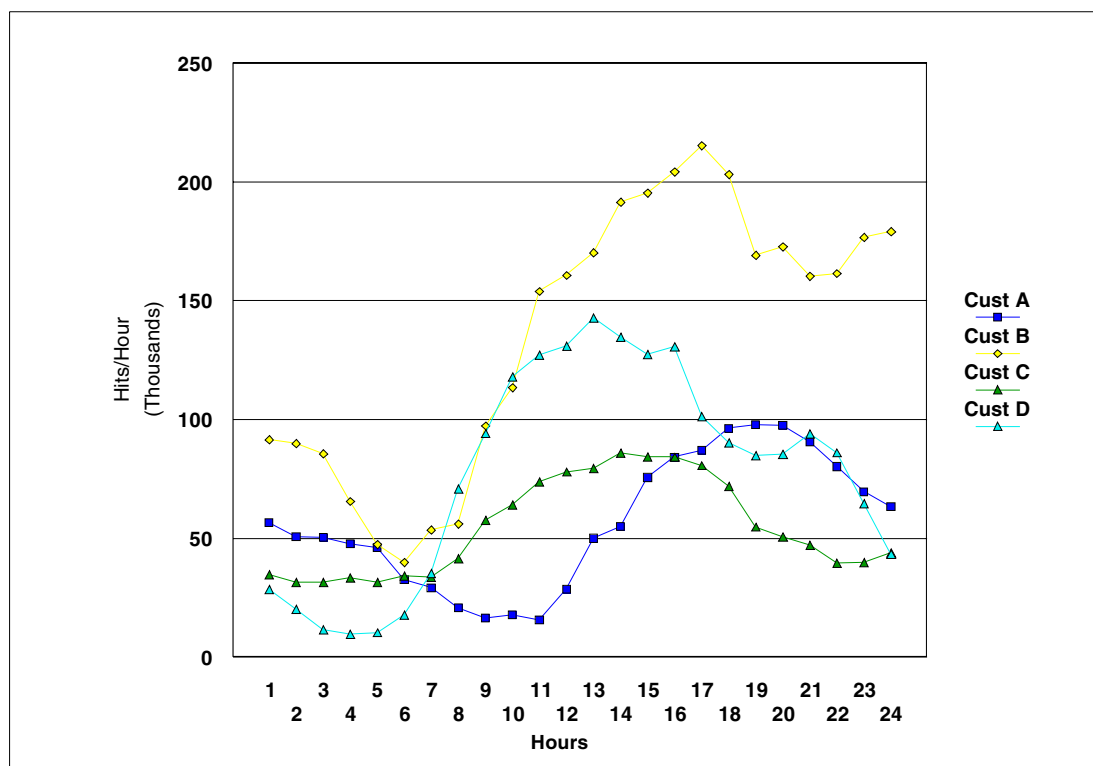


Figure 1-1 Some typical Web site loads over a 24-hour period

An example of a bursty site would be an online trading site at the opening of the market, or a sporting site during a very popular event. Such variation underscores the challenge of planning for volumes, and suggests that planning for average volumes is unlikely to be effective. Other factors that define the workload pattern include the volume of page views and transactions, the volume and type of searches, the complexity of transactions, the volatility of the data, and security considerations.

Figure 1-2 shows how seasonality can affect retail sites; notice the peak during the December holiday season and how it grew from 1999 to 2000.

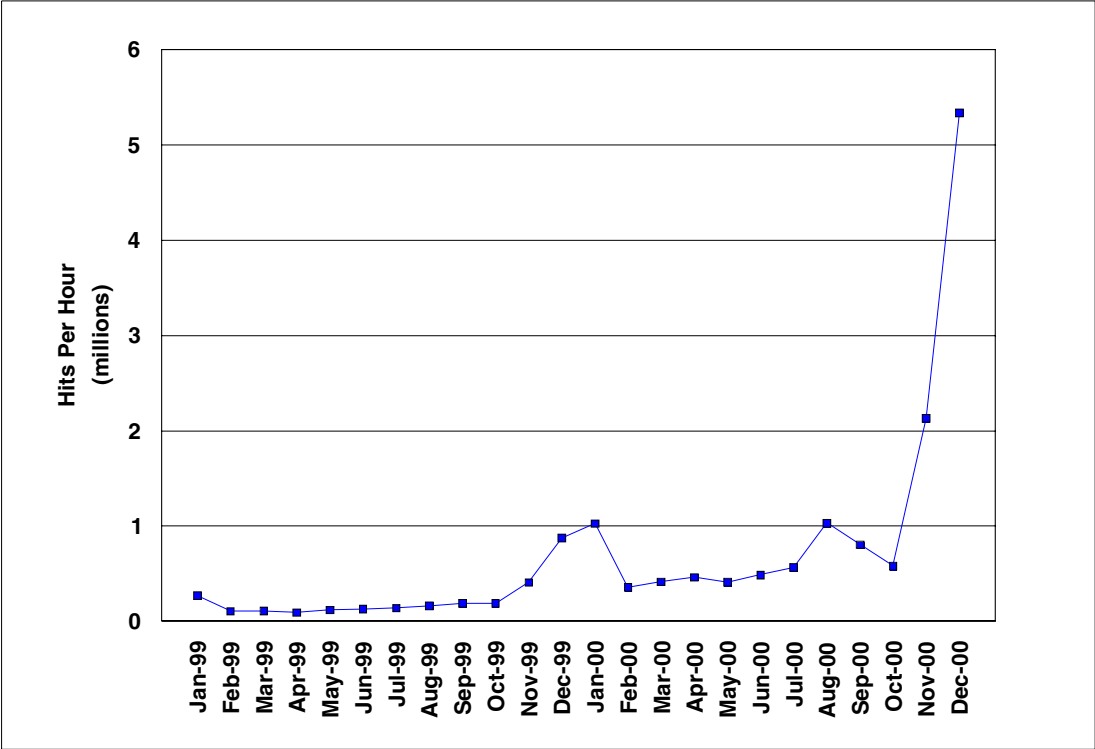


Figure 1-2 Example of a retail site with seasonal peaks, growing from year to year

Figure 1-3 is an example of the broad range of hits per day versus page views per day.

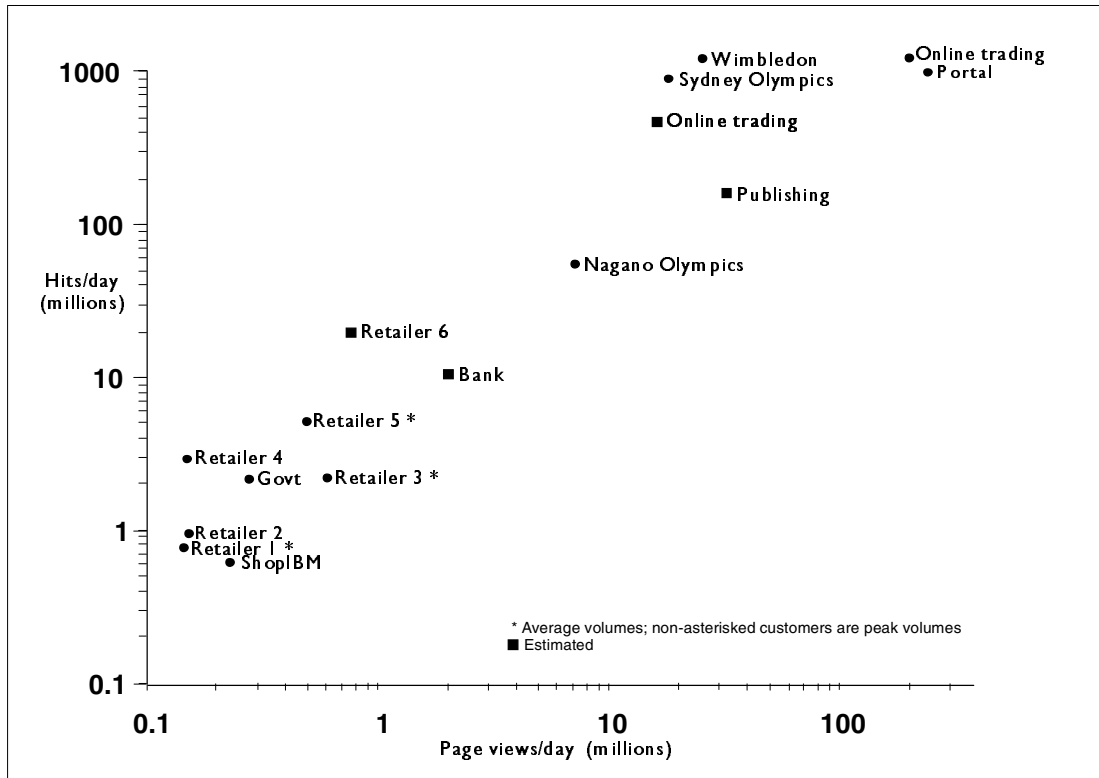


Figure 1-3 Examples of metrics for page hits per day

Introduction to workload patterns

Workload patterns vary, and sites with similar patterns can be classified into site types. We've identified five distinct workload patterns and corresponding Web site classifications. This section contains a guide for determining your workload pattern and selecting scaling techniques.

Publish/subscribe Web sites provide users with information

Sample publish/subscribe sites include search engines, media sites, such as weather.com and numerous newspapers and magazines, and event sites, such as those for the Olympics and the Wimbledon championships. Site content changes frequently, driving changes to page layouts. While search traffic is low in volume, the number of unique items sought is high resulting in the largest number of page views of all site types. As an example, the Sydney Olympics site successfully handled a peak volume of 1.2 million hits per minute using IBM's WebSphere Edge Server. The Wimbledon 2000 site successfully handled a peak volume of 430,000 hits per minute using IBM's WebSphere Edge Server. The Wimbledon 2001 site handled 208.5 million page views, three times the number of the 2000 site, as well as almost twice the number of unique users. Security considerations are minor compared to other site types. Data volatility is low. This site type processes the fewest transactions and has little or no connection to legacy systems.

Online shopping sites let users browse and buy

Sample sites include typical retail sites where users buy books, clothes, and even cars. Site content can be relatively static, such as a parts catalog, or dynamic where items are frequently added and deleted as, for example, promotions and special discounts that come and go. Search traffic is heavier than the publish/subscribe site, though the number of unique items sought is not as large. Data volatility is low. Transaction traffic is moderate to high, and almost always grows. The typical daily volumes for many large retail customers running on IBM's WebSphere Commerce Suite range from less than one million hits per day to over 50 million hits per day, with a range from 100,000 transactions per day to three million transactions per day for the higher-volume sites; of the total transactions, typically between 1% and 5% are buy transactions. When users buy, security requirements become significant and include privacy, nonrepudiation, integrity, authentication, and regulations. Shopping sites have more connections to legacy systems, such as fulfillment systems, than the publish/subscribe sites, but generally less than the other site types.

Customer self-service sites let users help themselves

Sample sites include banking from home, tracking packages, and making travel arrangements. Home banking customers typically review their balances, transfer funds, and pay bills. Data comes largely from legacy applications and often comes from multiple sources, thereby exposing data consistency. Security considerations are significant for home banking and purchasing travel services, less so for other uses. Search traffic is low volume; transaction traffic is moderate, but growing rapidly.

Trading sites let users buy and sell

Of all site types, trading sites have the most volatile content, the highest transaction volumes (with significant swing), the most complex transactions, and are extremely time sensitive. Auction sites are characterized by highly dynamic bidding against items with predictable life times. Products like IBM's WebSphere Application Server have the performance features that enable these sites to meet customer demand. Trading sites are tightly connected to the legacy systems, for example, using IBM's MQSeries® for connectivity. Nearly all transactions interact with the back end servers. Security considerations are high, equivalent to online shopping, with an even larger number of secure pages. Search traffic is low volume.

Using Web services, B2B sites buy from/sell to each other

These sites include dynamic programmatic links between arms-length businesses (where a trading partner agreement might be appropriate). One business is able to discover another business with which it may want to initiate transactions. Example: supply chain management. Data comes largely from legacy applications and often comes from multiple sources, thereby exposing data consistency. Security requirements are equivalent to online shopping. Transaction volume is moderate, but growing; transactions are typically complex, connecting multiple suppliers and distributors.

Understanding your workload

Here's a summary of the five HVWS workload patterns and in Tables 1-1, 1-2, and 1-3 you may wish to use the information when categorizing your Web site and determining which characteristics most affect the components of your infrastructure.

Table 1-1 Workload patterns and Web site classifications

Site Type	Publish/ Subscribe	Online Shopping	Customer Self-Service	Trading	Web Services/ B2B
Pattern					
Categories/ Examples	Search engines Media Events	Exact inventory Inexact inventory	Home banking Package tracking Travel arrangements	Online stock trading Auctions	eProcurement
Content	Dynamic change of the layout of a page, based on changes in content, or need Many page authors and page layout changes frequently High volume, non user specific access Fairly static information sources	Catalog either flat (parts catalog) or dynamic (items change frequently, near real time) Few page authors and page layout changes less frequently User specific information: user profiles with data mining	Data is in legacy applications Multiple data sources, requirement for consistency	Extremely time sensitive High volatility Multiple suppliers, multiple consumers Transactions are complex and interact with back end	Data is in legacy applications Multiple data sources, requirement for consistency Transactions are complex
Security	Low	Privacy, nonrepudiation, integrity, authentication, regulations	Privacy, nonrepudiation, integrity, authentication, regulations (Banking); Low for others	Privacy, nonrepudiation, integrity, authentication, regulations	Privacy, nonrepudiation, integrity, authentication, regulations
Percent Secure Pages	Low	Medium	Medium	High	Medium

Site Type	Publish/ Subscribe	Online Shopping	Customer Self-Service	Trading	Web Services/ B2B
Cross-session Info	No	High	Yes	Yes	Yes
Searches	Structured by category Totally dynamic Low volume	Structured by category Totally dynamic High volume	Structured by category Low volume	Structured by category Low volume	Structured by category Low to moderate volume
Unique Items	High	Low to Medium	Low	Low to Medium	Moderate
Data Volatility	Low	Low	Low	High	Moderate
Volume of transactions	Low	Moderate to High	Moderate and growing	High to Very High (very large swings in volume)	Moderate to Low
Legacy Integration/Complexity	Low	Medium	High	High	High
Page Views	High to Very High	Moderate to High	Moderate to Low	Moderate to High	Moderate

If your application has a characteristic not included in Table 1-2, add it if you believe scalability would be affected.

Table 1-2 Characterizing your workload

Site Type	Publish/ Subscribe	Online Shopping	Customer Service	Trading	Web Services/ B2B	Your Workload
Character of Workload						
High Volume Dynamic Transactional Fast Growth	Yes	Yes	Yes	Yes	Yes	Yes
Volume of User specific Responses	Low	Low	Medium	High	Medium	
Amount of Cross Session Information	Low	High	High	High	High	
Volume of Dynamic Searches	Low	High	Low	Low	Medium	
Transaction Complexity	Low	Medium	High	High	High	

Site Type	Publish/ Subscribe	Online Shopping	Customer Service	Trading	Web Services/ B2B	Your Workload
Transaction Volume Swing	Low	Medium	Medium	High	High	
Data Volatility	Low	Low	Low	High	Medium	
Number of Unique Items	High	Medium	Low	Medium	Medium	
Number of Page Views	High	Medium	Low	Medium	Medium	
Percent Secure Pages (privacy)	Low	Medium	Medium	High	High	
Use of Security (authenticate, integrity, nonrepudiate)	Low	High	High	High	High	
Other Workload Character Items						High

Table 1-3 is generalized and may not match your specific workload. Remember, if you added a characteristic to the previous table you will need to add that characteristic to this table as well.

Table 1-3 Determine the components most affected

Infrastructure Component	Edge Server	Web Presentation Server	Web Application Server	Security Servers	Transaction Servers	Data Servers	Network
Character of Workload							
High volume, dynamic, transactional, fast growth	High	High	High	High	High	High	High
High percent user traffic responses	Low	Low	High	Low	High	High	Low
High percent cross session information	Low	Medium	High	Low	Low	Medium	Low

Infrastructure Component	Edge Server	Web Presentation Server	Web Application Server	Security Servers	Transaction Servers	Data Servers	Network
High volume of dynamic searches	Medium	High	High	Low	Medium	High	Medium
High transaction complexity	Low	Low	High	Medium	High	High	Low
High transaction volume (swing)	Low	Medium	High	Low	High	High	Low
High data volatility	Low	High	High	Low	Medium	High	Low
High number unique items	Low	Low	Medium	Low	High	High	Low
High number page views	High	High	Low	Low	Low	Low	High
High percent secure pages (privacy)	Low	High	Low	High	Low	Low	Low
High security	Low	High	High	High	High	Low	Low



Designing for scalability

The significance of designing for scalability and high performance cannot be understated. We know from analysts that slow performance costs e-business sites many millions of dollars per month. Volumes are growing as well. For example, the 2001 Wimbledon site had nearly twice the number of unique users, and three times the number of page views as the 2000 site. An e-commerce site we work with saw their holiday season peak hits grow from just under 1 million hits per hour in 1999 to over 5 million hits per hour in 2000. We work with an online trading site whose page views per day grew from approximately nine million to approximately 16 million over the same period. Page views per day at an auction site grew from approximately 65 million to approximately 200 million over the same period.

You know that the success of your company's e-business depends on your organization's ability to design and implement an infrastructure that yields the measures of high performance, availability, and reliability expected to support the business objectives for revenue and customer satisfaction. The infrastructure consists of hardware, software, and network components you select for their ability to meet your needs of today and tomorrow.

Scalability refers to a component's ability to adapt readily to a greater or lesser intensity of use, volume, or demand while still meeting business objectives. Understanding the scalability of the components of your e-business infrastructure and applying appropriate scaling techniques can greatly improve availability and performance. Scaling techniques are especially useful in multi-tier architectures when you evaluate components associated with the edge servers, the Web presentation servers, the Web application servers, and the data and transaction servers.

The objective of this chapter is to introduce scalability and scaling techniques and to help you understand that to optimize for the success of your company's e-business, you must evaluate all new and upgraded components of your infrastructure for scalability. IBM's IT experts have been working with customers to analyze many of the world's largest Internet and intranet sites, including IBM's own, to determine which attributes affected scalability and to help customers implement scalable Web sites. This chapter will help you understand your workload patterns and classify your site. You'll learn which scaling techniques are best for specific components and how other large customers have realized the benefits of scaling using IBM's middleware products, such as WebSphere®, MQSeries®, DB2®, and Tivoli®.

Introducing scalability

Most e-businesses face the same challenges, the most significant of which are unpredictable growth and the ability to have solutions ready for unknown problems. If your Web site is typical, it most likely started with displaying company information and has evolved to processing simple, if not, complex transactions. You know now that the skills required to display information are different from those required to process transactions. Failure to optimize graphics, frequent table scans and joins of multiple tables, and the resulting I/O bottlenecks combine to degrade performance. Site availability is stressed by unpredictable traffic and inadequate discipline regarding systems management. The problems you're facing may be compounded by poor application design and systems that are poorly configured, under powered, or both.

Meeting such challenges requires unprecedented flexibility and capacity for change in all areas, especially your IT operation. We believe the success of future e-businesses may be tied to the selection of components that can be individually and/or collectively adjusted to meet variable demands. Such flexibility is called scalability and is a feature your team needs to understand and measure for each component within your infrastructure. Scalability is related to the features of performance (response time) and capacity (operations per unit of time) but should not be considered synonymous.

Scaling a multi-tiered infrastructure from end to end means managing the performance and capacities of each component within each tier. The basic objectives of scaling a component/system are to:

- ▶ Increase the capacity or speed of the component
- ▶ Improve the efficiency of the component/system
- ▶ Shift or reduce the load on the component.

As one increases the scalability of one component, the result may change the dynamics of the site service, thereby moving the "hot spot" or bottleneck to another component. The scalability of the infrastructure depends on the ability of each component to scale to meet increasing demands. Figure 2-1 illustrates the relationship between performance curves, response time, and the scaling target.

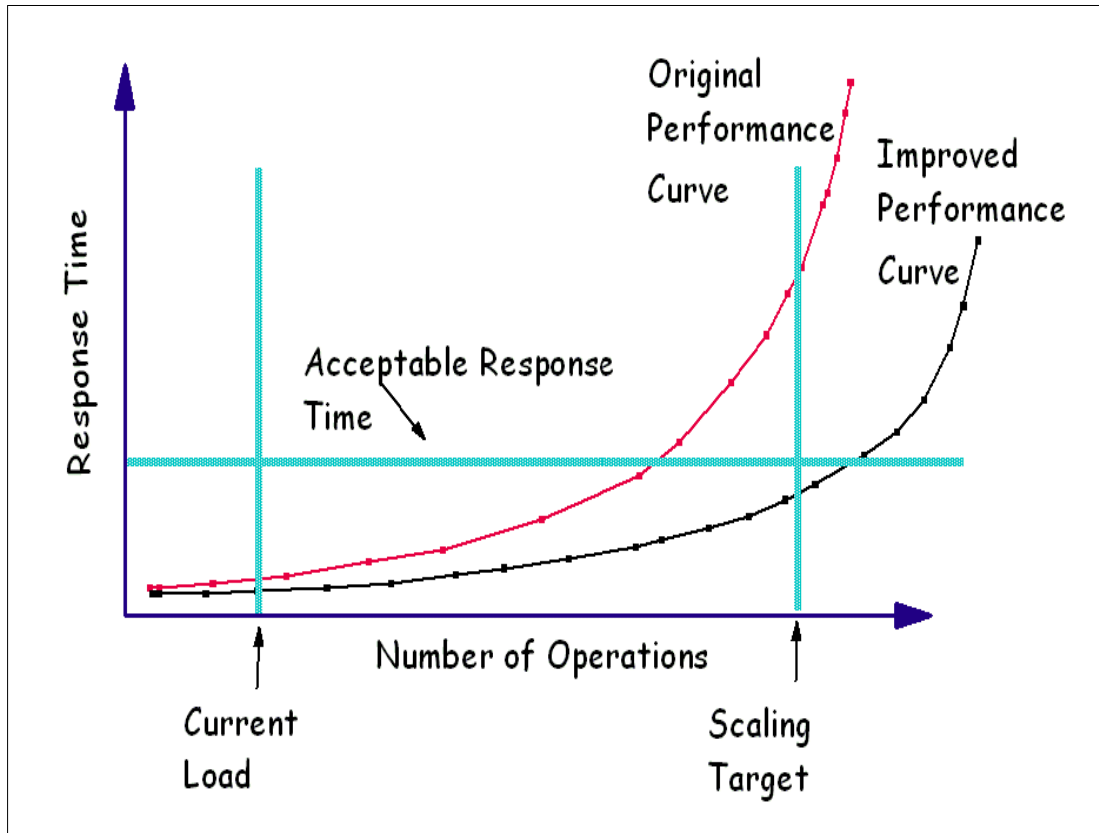


Figure 2-1 Scalability/performance curves

Notice the "Original Performance Curve" is unable to meet the acceptable level of response time for the scaling target. By scaling components within a Web site, we can improve the performance of the site. Refer to the Improved Performance Curve in Figure 2-1.

Table 2-1 introduces the scaling techniques and relates them to the scaling objectives. For example, if your objective is to increase the speed of a component, you would consider using a faster or special machine and/or creating a machine cluster. Altering the load on a component is often less straightforward. For example, an item in a cache can be served up faster than an item in a database. If a large number of requests can be handled using a cache instead of the database, the overall load on the database is reduced, affording greater scalability for the entire system. Frequently, the techniques that reduce load on one component actually make other components more efficient, thus compounding the scaling effect.

Table 2-1 How scaling techniques relate to scaling objectives

ID	Scaling Technique	Increase Capacity/Speed	Improve Efficiency	Shift/Reduce Load
1	Use faster machine	X		
2	Create machine cluster	X		
3	Use a special machine	X	X	
4	Segment the workload		X	X
5	Batch requests		X	

ID	Scaling Technique	Increase Capacity/Speed	Improve Efficiency	Shift/Reduce Load
6	Aggregate user data		X	
7	Manage connections		X	
8	Cache data and requests		X	X

Six steps to scaling your infrastructure

We recommend you consider this high-level approach to classifying your Web site and learning which scaling techniques could be applied. The approach is systematic, but you and your best IT architects will need to improvise and adapt the approach to your situation. There are six steps:

1. Understand the application environment
2. Categorize your workload
3. Determine the components most impacted
4. Select the scaling techniques to apply
5. Apply the techniques
6. Reevaluate

Knowing your workload pattern (publish/subscribe and customer self-service, for example) determines where to focus your scalability efforts, and which scaling techniques to apply. For example, a customer self-service site such as an online bank needs to focus on transaction performance, and the scalability of databases that contain customer information used across sessions. These considerations would not typically be significant to a publish/subscribe site.

Step 1. Understand the application environment

For existing environments, the first step is to identify all components and understand how they relate to each other. The most important task is to understand the requirements and flow of the existing application(s) and what can or cannot be altered. The application is key to the scalability of any infrastructure, so a detailed understanding is mandatory to scale effectively. At a minimum, your analysis must include a breakdown of transaction types and volumes as well as a graphic view of the components in each tier.

Figure 2-2 can help you determine where to focus your scalability planning and tuning efforts. The figure shows where latency is greatest for representative customers in three of the workload patterns, and in which tier you should concentrate for each. For example, for online banking, most of the latency typically occurs in the database server, whereas the application server typically experiences the greatest latency for online shopping and trading sites. The way applications manage traffic between tiers significantly affects the distribution of latencies between the tiers, which suggests that careful analysis of application architectures is an important part of this step and could lead to reduced resource utilization and faster response times. You should collect metrics for each tier, and make behavior predictions for your users for each change you implement. WebSphere Application Server has application programming interfaces that provide detailed data useful for monitoring application performance.

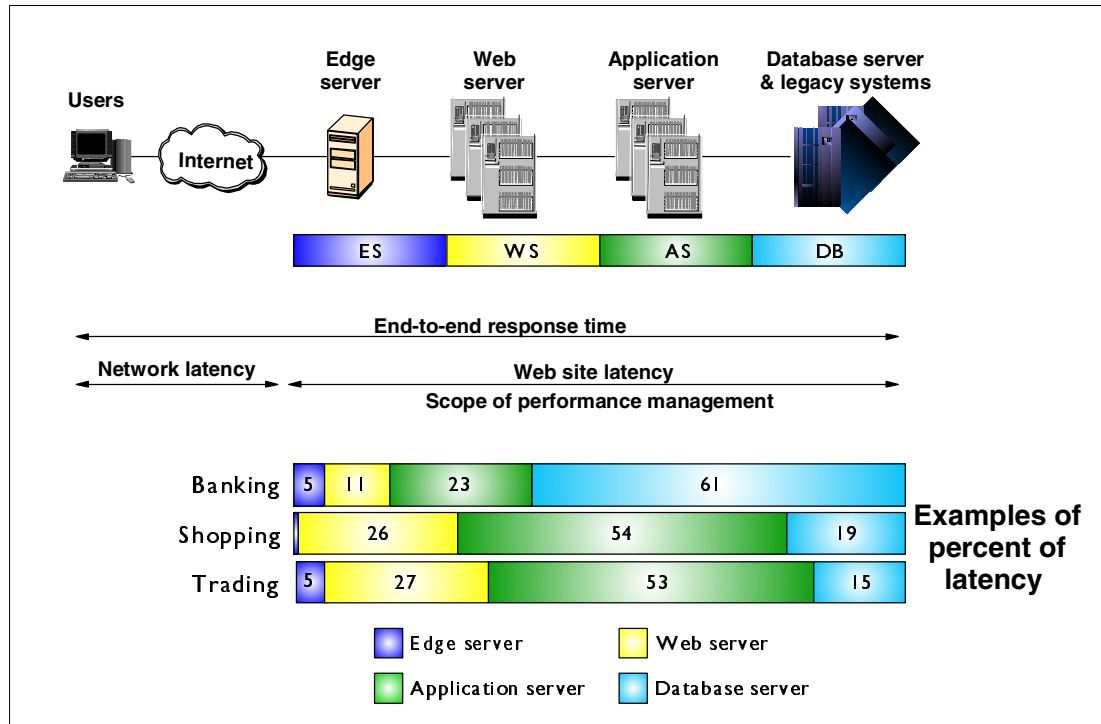


Figure 2-2 How latency varies based on workload pattern and tier

As you analyze requirements for a new application, you have the opportunity to build scaling techniques into your infrastructure. New applications offer you the opportunity to consider all that is new in the areas of each component type, such as open interfaces and new devices, the potential to achieve unprecedented transaction rates, and the ability to employ rapid application development practices. Each technique affects application design; similarly, application design impacts the effectiveness of the technique. To achieve proper scale, application design must consider potential scaling effects. In the absence of known workload patterns, you'll need to follow an iterative, incremental approach.

Step 2. Categorize your workload

All site types, like yours, are considered to have high-volumes of dynamic transactions. Your site type will become clear as you evaluate your site for the other characteristics that pertain to transaction complexity, volume swings, data volatility, security, and others. If you need help, refer to Table 1-1 on page 7 and Table 1-2 on page 8 in Chapter 1.

This step involves mapping the most important site characteristics to each component. Once again, from a scalability viewpoint, the key components of the infrastructure are the edge servers, the Web application servers, security services, transaction and data servers, and the network. Table 1-3 on page 9 in Chapter 1 specifies the significance of each workload characteristic to each component. As you can see, the affect on each component is different for each workload characteristic.

Step 3. Determine the components most affected

This step involves mapping the most important site characteristics to each component. Once again, from a scalability viewpoint, the key components of the infrastructure are the edge servers, the Web application servers, security services, transaction and data servers, and the network. Table 1-3 on page 9 in Chapter 1 specifies the significance of each workload characteristic to each component. As you can see, the effect on each component is different for each workload characteristic.

Step 4. Select the scaling techniques to apply to scale the workload

It is worth the best efforts of your IT architects to collect the information needed to make the best scaling decision. Only when the information gathering is as complete as it can be is it time to consider matching scaling techniques to components. Manageability, security, and availability are critical factors in all design decisions. Techniques that provide scalability but compromise any of these critical factors cannot be used.

Here's a summary of the eight scaling techniques.

1. **Use a faster machine** This technique applies to the edge servers, the Web presentation server, the Web application server, the directory and security servers, the existing transaction and data servers, the network, and the Internet firewall. The goal is to increase the ability to do more work in a unit of time by processing tasks more rapidly. A faster machine can be achieved by upgrading the hardware or software. However, one of the issues is that software capabilities can limit the hardware exploitation and vice versa. Another issue is that due to hardware or software changes, changes may be needed to existing system management policies.
2. **Create a cluster of machines** This technique applies to the Web presentation server, the Web application server, and the directory and security servers. The primary goal here is to service more client requests. Parallelism in machine clusters typically leads to improvements in response time. Also, system availability is improved due to failover safety in replicas. The service running in a replica may have associated with it state information that must be preserved across client requests, and thus needs to be shared among machines. State sharing is probably the most important issue with machine clusters and can complicate the deployment of this technique. IBM WebSphere's workload balancing feature uses an efficient data sharing technique to support clustering. Issues such as additional system management for hardware and software can also be challenging.
3. **Use appliance servers** This technique applies to the edge servers, the Web presentation server, the directory and security servers, the network, and the Internet firewall. The goal is to improve the efficiency of a specific component by using a special purpose machine to perform the required action. These machines tend to be dedicated machines that are very fast and optimized for a specific function. Examples are network appliances and routers with cache, such as the IBM WebSphere Edge Server. Our experience with the Wimbledon 2001 Web site demonstrated tremendous benefits by using caching; it can reduce up to 85% of the HTTP traffic to the presentation servers. Some issues to consider regarding special machines are the sufficiency and stability of the functions and the potential benefits in relation to the added complexity and manageability challenges. It's worth noting, however, that the newer generation of devices are increasingly easy to deploy and manage; some are even self-managed.
4. **Segment the workload** This technique applies to the Web presentation server, the Web application server, the data server, the intranet firewall, and the network. The goal is to split up the workload into manageable chunks thereby obtaining more consistent and predictable response time. The technique also makes it easier to manage which servers the workload is being placed on. Combining segmentation with replication often offers the added benefits of providing an easy mechanism to redistribute work and scale selectively as business needs dictate. An issue with this technique is that in order to implement the segmentation, one needs to be able to characterize the different workloads serviced by the component. After segmenting the workload, additional infrastructure is required to balance physical workload among the segments, for example, the use of the IBM WebSphere Edge Server.
5. **Batch requests** This technique applies to the Web presentation server, the Web application server, the directory and security servers, the existing business applications, and the database. The goal is to reduce the number of requests sent between requesters and responders (such as between tiers or processes) by allowing the requester to define

new requests that combine multiple requests. The benefits of this technique arise from the reduced load on the responders by eliminating overhead associated with multiple requests. It also reduces the latency experienced by the requester due to the elimination of the overhead costs with multiple requests. Some of the issues are that there may be limits in achieving reuse of requests due to inherent differences in various requests types (such as Web front end differs from voice response front end). This can lead to increased costs of supporting different request types.

6. **Aggregate user data** This technique applies to the Web presentation server, the Web application server, and the network. The goal is to allow rapid access to large customer data controlled by existing system applications and support personalization based on customer specific data. When accessing existing customer data spread across existing system applications, the existing applications may be overloaded, especially when the access is frequent. This can degrade response time. To alleviate this problem, the technique calls for aggregating customer data into a customer information service (CIS). A CIS that is kept current can provide rapid access to the customer data for a very large number of customers; thus, it can provide the required scalability. An issue with a CIS is that it needs to scale very well to support large data as well as to field requests from a large number of application servers (requesters).
7. **Manage connections** This technique applies to the Web presentation server, the Web application server, and the database. The goal is to minimize the number of connections needed for an end-to-end system, as well as to eliminate the overhead of setting up connections during normal operations. To reduce the overhead associated with establishing connections between each layer, a pool of preestablished connections is maintained and shared among multiple requests flowing between the layers. For instance, most application servers provide database connections managers to allow connection reuse. It is important to note that a session may use multiple connections to accomplish its tasks, or many sessions may share the same connection. This is called connection pooling in the WebSphere connection manager. The key issue is with maintaining a session's identity when several sessions share a connection. Reusing existing database connections conserves resources and reduces latency for application requests, thereby helping to increase the number of concurrent requests that can be processed. Managing connections properly can improve scalability and response time. Administrators must monitor and manage resources proactively to optimize component allocation and use.
8. **Cache** Caching is a key technique to reduce hardware and administrative costs and to improve response time. Caching applies to the edge server, the Web presentation server, the Web application server, the network, the existing business applications, and the database. The goal is to improve the performance and scalability by reducing the length of the path traversed by a request and the resulting response, and by reducing the consumption of resources by components. Caching techniques can be applied to both static and dynamic Web pages. A powerful technique to improve performance of dynamic content is to asynchronously identify and generate Web pages that are affected by changes to the underlying data. Once these changed pages are generated, they must then be effectively cached for subsequent data requests. There are several examples of intelligent caching technologies that can significantly enhance the scalability of e-business systems. The key issue with caching dynamic Web pages is determining what pages should be cached and when a cached page has become obsolete.

Rather than buying hardware that can handle exponential growth that may or may not happen, consider specific approaches for these two types of servers:

- For application servers, the main technique for growth path is to add more machines. It is therefore appropriate to start with the expectation of more than one application server with a dispatcher in front, such as IBM's WebSphere Edge Server. Adding more machines then becomes painless and far less disruptive.

- For data servers, get a server that is initially oversized; some customers run at just 30% capacity. This avoids the problem in some environments where the whole site can only use one data server. Another scaling option when more capacity is needed is to partition the database into multiple servers.

Most sites we studied separate the application server from the database server. They place front-end Web serving and commerce application functions on less expensive, commodity machines and the data server on more robust and secure but more expensive systems. The trend with many publish/subscribe sites is to put the Web server on eServer pSeries or PCs and to put the application and databases on larger systems such as high-end pSeries or zSeries systems. In many accounts, the most important performance tuning factor becomes the data server. Many commerce sites do not have large databases and achieve improved performance by caching most or all of the databases in memory.

Step 5. Apply the technique(s)

Apply the selected technique(s) in a test environment first to evaluate not only the performance / scalability impact to a component, but also to determine how each change affects the surrounding components and the end-to-end infrastructure. You do not want a situation where improvements in one component result in an increased (and unnoticed) load on another component. Figure 2-3 illustrates the typical relationship between the techniques and the key infrastructure components. By using this figure, you can identify the key technique for each component. In many cases, all techniques cannot be applied because one or more of the following is true:

- You cannot afford to invest in the techniques, even if it would help.
- You won't perceive the need to scale as much as the techniques will provide.
- Your cost / benefit analysis shows that the technique will not result in a reasonable payback.

The IT architect must therefore have a process for applying these techniques in different situations so that the best return is achieved. This mapping is a starting point and shows the components to focus on first based on your workload.

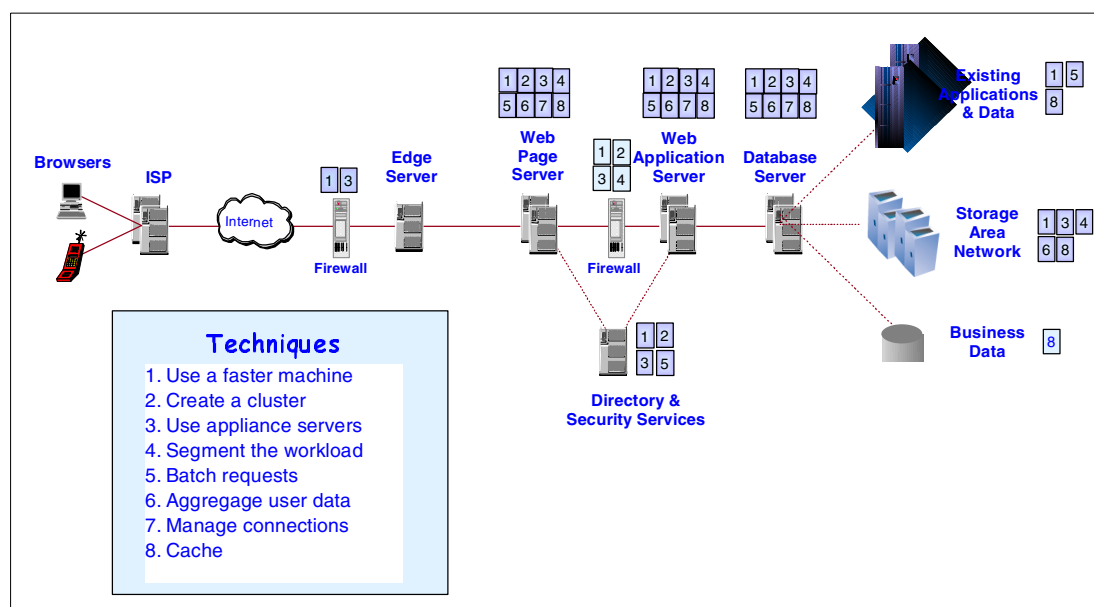


Figure 2-3 Scaling techniques applied to components

Step 6. Reevaluate

As with all performance related work, tuning will be required. The goals are to eliminate bottlenecks, scale to a manageable status those that can't be eliminated, and work around those that can't be scaled. One of IBM's large customers scaled its peak load 34 times, and achieved improved peaks as high as 12 million hits per hour using a combination of these techniques. Some of their tuning actions and the benefits realized were:

- ▶ Increasing Web server threads raised the number of requests that could be processed in parallel.
- ▶ Adding indexes to the data server reduced I/O bottlenecks.
- ▶ Changing defaults for several operating system variables allowed threaded applications to use more heaps.
- ▶ Caching significantly reduced the number of requests to the data server.
- ▶ Increasing the number of edge/appliance servers improved load balancing.
- ▶ Upgrading the data server increased throughput.

Such results demonstrate the potential benefits of systematically applying scaling techniques and continuing to tune. Figure 2-4 shows performance data from a recent scaling study for a Web site that uses IBM WebSphere Application Server.

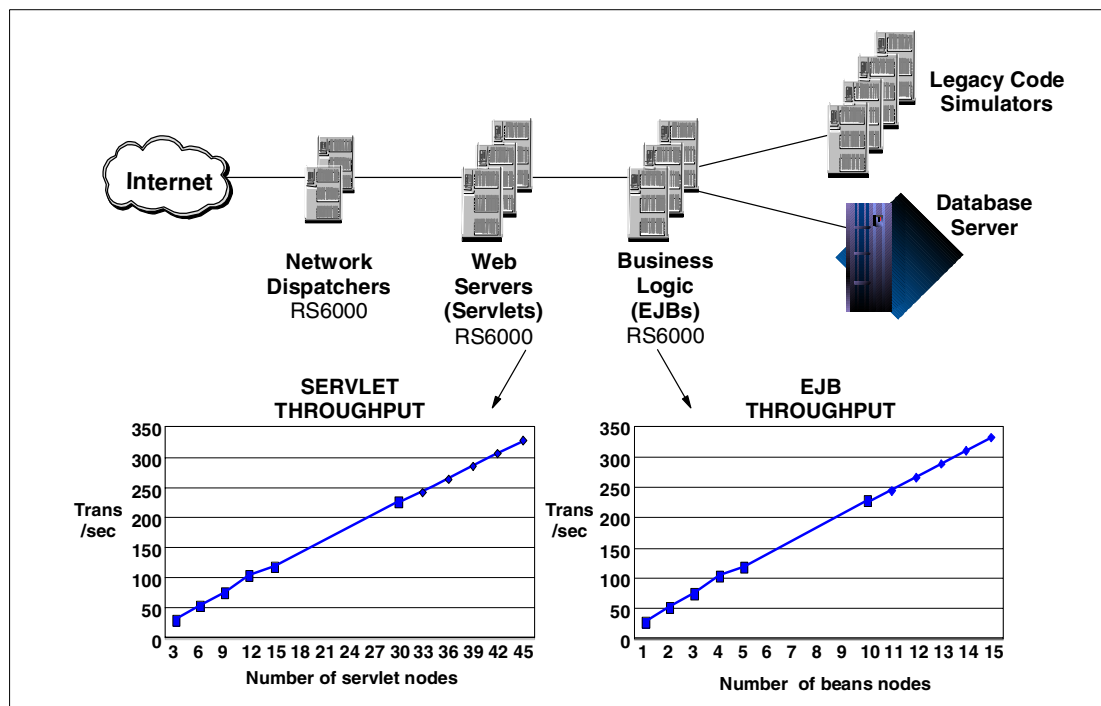


Figure 2-4 Scaling a WebSphere online trading site

Additional techniques

One of the most important techniques for high-volume sites with complex transaction workloads is to push as much work as possible toward the network. This allows the back end server to be tuned to handle critical transaction workload. Horizontally-scaled commodity servers can be used for page serving. An effective caching strategy can improve nearly every site's scalability.

IBM implemented a reusable infrastructure for its sporting event sites. Many practices have been refined over time. Caching is extensive. Scores are built into static pages that can be cached in the network, either in routers or ISP caches. When the scores change, the affected pages are dynamically updated and the caches invalidated. In general, the use of caching provides the quickest response time and lowers the system load since the network cached pages can be served directly to users without a hit to the Web server. Other practices considered most significant are flattening of product catalogs and fast-path shop design. This is the approach used for the product catalogs. Catalog pages are flattened into static Web pages that can be cached like other static pages for the event site. For every 100 browsers on the site, only 1 to 5 actually buy anything. 95% of the time, browsing is done without accessing the catalog database. The frequently accessed pages are even cached out in the network so the Web server load is lower too.

Managing runtime performance has special challenges. More than ever, this task requires a perspective that considers components from the front-end browsers to the back-end database servers. The end-to-end perspective must be understood and shared by the IT management, operations staff, application developers, and Web site designers. Also required are stated business goals, thoughtful performance objectives, and thorough performance measurements.

We've developed an end-to-end methodology that aligns the system performance with the underlying business goals. The methodology combines proactive monitoring, analysis, alert, and predictive mechanisms. To read about our end-to-end performance methodology, see the HVWS white paper *Managing Web Site Performance*.

An e-business must plan for success and look to the future. IBM developed the High-Volume Web Site (HVWS) Simulator to estimate the performance of complex configurations. The HVWS Simulator is an analytic queuing model that estimates the performance of a Web server based on workload patterns, performance objectives, and specified hardware and software. The results can be used as guidelines for configuration sizing. Performance results are displayed in sufficient detail to allow users to assess the adequacy of a given configuration for their requirements, and to provide insight into where the bottlenecks are likely to occur. This allows the simulator to be useful for capacity planning, evaluation of infrastructure and workload changes, projecting Web site scalability, and reducing the cost of prototyping.

Through monitoring and adding traffic load predictions with capacity estimates, we can achieve a level of automation for self-managed systems. A self-managed system should be able to handle a wide variety of workload patterns with satisfactory performance. They should be able to recover gracefully from the stress of unexpected workloads. Therefore, it is important to monitor system performance in real time, collect necessary data, and feed that data to predictive tools that can predict workload increases ahead of time. WebSphere Application Server now has application programming interfaces that provide detailed data useful for monitoring the performance of applications.

Finally, managing quality of service can increase the availability of resources to prioritized requests. This involves allocating limited sharable resources to the requests that need them most by classifying and qualifying requests based on business policies. For example, you may classify users who buy or are paying ahead of users who browse or are searching. This technique is directly applicable to online shopping (registered shoppers or frequent shoppers), self-service, and online trading sites. You may also distinguish between requests based on complexity. For example, separate the typical simple transactions (product display) from the complex (payment authorization, tax/shipping/discount calculations).

Common pitfalls

For over two years now, the HVWS team has been working closely with customers to analyze their e-business infrastructures across all phases of their life cycles, from planning and architecture through development, test, and deployment. This section summarizes the most common mistakes customers make when implementing their e-business infrastructures.

Planning phase

Common planning phase mistakes are:

Customers often have weak linkages between their business planning and IT shops, resulting in poorly defined scalability targets and growth plans. For example, when a business team plans a major ad campaign, it must give the IT organization sufficient notice to prepare the infrastructure to meet demand. One customer experienced what we call "e-panic" to rapidly install 26 new T1 lines to handle a special promotion where they originally thought one T1 line would suffice.

Customers sometimes fail to scale their business processes as well as their e-business infrastructure. We worked with a "click and mortar" business that couldn't deliver holiday season goods bought through their Web site because their inventory was insufficient to fulfill their total orders. One health insurance company wisely deferred deployment of an e-business application until its back office business process could be scaled to anticipated volumes.

Architecture phase

Common architecture phase mistakes are:

Some IT shops try to "reinvent the wheel." Instead of focusing on their business problem and their applications, they develop and implement e-business middleware. We've seen homegrown workload balancers, HTTP servers, and Web application servers. This can introduce major cost, maintainability, and scalability issues. With the popularity of J2EE standard platforms, such as WebSphere, reinventing occurs less frequently.

Customers who don't design for scalability limit their options when volumes increase. It is difficult to project volumes, and because volumes can grow quickly, it is best to have a scalable architecture that can cope with unexpected growth. One large customer launched a new site and, on the first day, exceeded its projection for the first year. Many customers have had reasonable success by projecting new site traffic based on existing site traffic; others have found projecting traffic based on business expectations to be effective. We recommend that you plan for workload balancing, and deploy a workload balancer from day one, even if you only start with a single server. You can add capacity easily when growth occurs.

Design phase

A common design phase mistake is:

Customers don't design Web pages with a performance objective. The latest consultant studies indicate that many consumers are unwilling to wait more than eight seconds for a page to download. Well-designed Web pages load faster and enable a site to handle more users.

Development and test phases

The key point about these phases is that there are no shortcuts in the e-business environment. In fact, due to the complexities of the multi-tier infrastructure, and the external visibility of performance and availability problems, even more rigor is required.

Many customers struggle to adhere to a development and test process that provides adequate quality on the aggressive schedules demanded by the Internet. Too often we see compromises in the testing phase, especially in stress testing. Stress testing is a powerful tool in ensuring the scalability of a Web site, and in identifying and fixing bottlenecks that inhibit scaling.

We have tested scalability with one of our online trading customers to ensure that their new, J2EE-based architecture would scale to very high volumes. We conducted a recent scalability test with a bank, including testing connected to their production data servers. This proved to have unique value in that it identified bottlenecks typically not found in a controlled test environment, for example production firewall settings and limitations in bandwidth to the data servers.

Deployment phase

Common mistakes in the deployment phase are:

Inadequate change control procedures cause site outages and slowdowns. In one instance, a bank's customers' balances were visible to others due to a site applying an untested application change. In another case, configuration changes at a bank resulted in limits being set on the number of users able to access the system through a firewall. Many customers were unable to log onto their accounts.

Both of these IT shops had been around long enough to appreciate the importance of change control, but had made unwise exceptions to their policies due to the time pressures and complexities of their Web environments.

Complex, multi-tier Web environments require cross-functional cooperation and processes. To manage availability and performance across a multi-tier, multiplatform environment, groups that may have operated independently in the past need to work closely. A customer transaction's availability and performance depends on networks, firewalls, and often both UNIX™ and mainframe servers. Operational procedures need to span all of these components, covering a complete end-to-end perspective.

Choosing between two and three tiers

Multi-tier e-business infrastructures provide opportunities for improved scalability and performance not found in the early days of e-business. These opportunities come with complexities and challenges that we are just beginning to understand. Figure 2-5 summarizes the variety of multi-tier infrastructures in use by many of IBM's large customers.

Most modern application architectures require flexibility and thus are usually divided into logical layers for presentation logic, business logic, and data serving. The infrastructure of most large sites is comprised of two or more physical tiers with application function distributed among participating servers. Depending on workload pattern, presentation and business logic may coreside in one tier, with data serving in a separate tier; or, each may have its own tier; thus creating a three-tier infrastructure. Scaling is easier with a multi-tier architecture. As customers scale for increased volumes, they are implementing their applications on the J2EE computing platform, which facilitates implementation of scalable multi-tier architectures.

At the highest level, we know that the two-tiered infrastructure is simple, performs well, and is easiest to maintain. In many two-tier implementations, customers place both presentation logic and business logic on the first tier. However, with just one firewall between the Internet and the business logic, the two-tiered design has potential security exposures. These can be mitigated by implementing a three-tier structure, moving the business logic to a middle tier behind another firewall. However, as each tier is added, there are scalability and performance considerations. In some cases, it may make sense to consolidate all logical layers into a single large system complex.

The optimal solution for your site depends on your workload characteristics and application environment. A publishing site such as weather.com has a lot of presentation logic and data, but not much business logic. An online trading site typically has moderate presentation logic and a lot of business and data logic. The flexibility of having multiple tiers and multiple servers creates opportunities, challenges, and complexities. WebSphere is exceedingly well suited to provide the flexibility and robustness needed for these multi-tier alternatives.

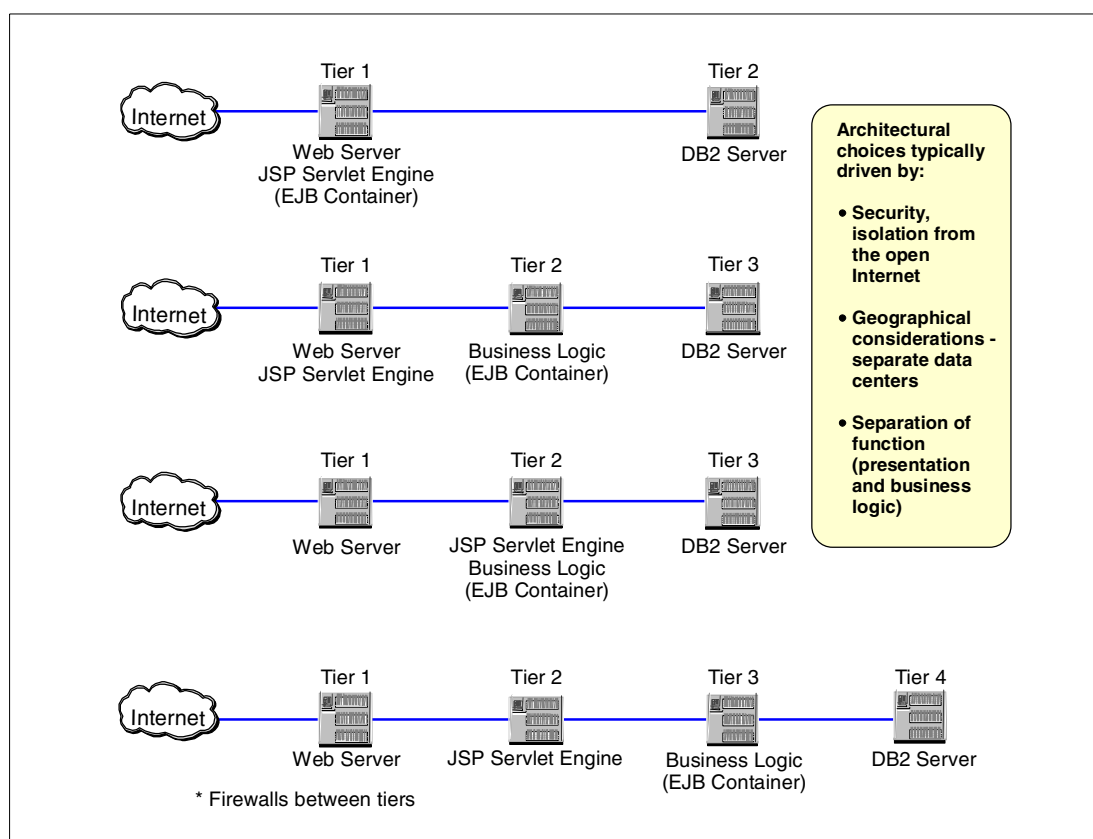


Figure 2-5 Examples of two-, three-, and four-tiered infrastructures

Emerging standards and technologies

Emerging standards and technologies will affect scalability, some in ways we are just starting to understand.

When scaling to new volumes, customers are implementing their Web applications using *J2EE* as the standard enterprise computing platform, which makes it easier to implement scalable multi-tier architectures. Customers are looking at *Linux* as a way to create more open solutions at reduced cost. Many are considering the implications of *Web services*. Web services is an evolving standard way to expose applications or resources on a network. It

consists of enabling technologies that allow processes and subprocesses to be located and accessed over the Web. Web services exploit e-business infrastructures by allowing them to focus on their core competencies and use partners to supply the remaining capabilities their business needs to operate. At a recent Web services show, 60% of the IT executives surveyed expect to use Web services for internal applications, such as retrieving customer information from a customer relationship management system or passing transactions from front-office systems to back-office systems. Customers are beginning to implement some of these standards. For example, XML is increasingly used for managing content and communicating between tiers. We recently implemented an architecture to access DB2 tables using Web services and DB2 XML extenders.

Self-managing servers are a key new IBM offering in response to our customers' challenge of managing their complex infrastructures comprised of multiple tiers and multiple servers. IBM's new eServer z900 and z/OS offerings will be able to reallocate processing power to a given application, based on the workload demands of the moment, and thus enabling tremendous capacity expansion and minimally disruptive scalability.

Pervasive computing devices such as cellular phones, Pads, and handheld computers are becoming prevalent around the world. In geographies such as the Far East and Europe, the wireless realm has been exploding. This area is also growing in the U.S., although at a more moderate pace. Consultants estimate that by 2003 wireless devices will represent 2 billion Internet access points, more than 60% of all Internet devices. Our customers are asking us how to best structure their applications to add support for new devices easily and with minimum disruption, and about the scalability of systems that will support millions of wireless users. The delivery of video and music to wireless devices is also being explored and its performance is being tested. This application will create some interesting performance and scalability challenges. IBM is participating in this exciting new area with the WebSphere Everyplace Server (WES). WebSphere Everyplace Server provides an integrated, robust solution designed to ease the entry into pervasive computing and facilitate future scaling as needed.

Summary

Designing, integrating, and managing scalable multi-tier infrastructures can be complex, challenging tasks. However, knowledge, techniques, standards, products, and best practices are increasingly available to help you. This chapter provides updated information on IBM's experiences with large customers who face these same challenging tasks, many of whom are enjoying the benefits that come with improved scalability and performance.

The techniques introduced in this chapter are:

- ▶ Use a faster machine
- ▶ Create a machine cluster
- ▶ Use an appliance server
- ▶ Segment the workload
- ▶ Batch requests
- ▶ Aggregate user data
- ▶ Manage connections
- ▶ Cache data and requests

A combination of clustered machines, connection management, and caching has enabled many large customers to scale their Web sites to handle peak loads. We've seen increases in peak loads as high as 34-fold, and with improved peaks as high as 12 million hits per hour. Other customers have seen 40-50% improvements in the download time for their Web pages by applying techniques directed at page performance.

While scaling end-to-end infrastructures is becoming more science than art, it still presents the best hope for responding to and managing unpredictable demands on your Web site. With thorough knowledge of such features of your current infrastructure as workload patterns, Web site components, skills, and budget, your IT architects should begin now to factor scalability considerations and scaling techniques into their plan for the future.

The challenges to architect a scalable infrastructure are many and real, and they keep coming, almost at Internet speed. The corresponding opportunities to use new devices and techniques are just as plentiful and real, as are the business opportunities for those who do it right. IBM has products and services that can help you implement the scalable infrastructure needed to make your company's e-business succeed inside and out.



Designing pages for performance

You only get one chance to make a first impression. The significance of this adage for your business is, or likely will be, increasingly associated with your Web site. Many factors contribute to the first impression of your Web site, but the one your visitors are most likely to notice, and remember, is how long it takes to download the home page. And, if it takes too long, your visitors may decide not to stay, not to return, and not to tell their friends about you, at least not to tell them anything good. This is true regardless of the characteristics of your site, but its significance is great if your site is an e-commerce site.

Can you confirm the business value of each item on your site's home page? You want the equivalent of a budget for each page, where space is allocated to each item and you define the business value you expect returned for that item. Any item without business value degrades the time needed to download your Web page. Do you design and test your Web site in the environment most like the environment of the majority of your visitors? If most of your visitors connect to your site from a typical dial-in connection, your design must optimize for them to assure their visit is successful by either yielding business for you or, at least, a return visit from them. These are just two considerations when optimizing Web page design. More are discussed later.

IBM's High-Volume Web Site team has been working with customers to analyze many of the world's largest Internet and intranet sites. A portion of this work has been evaluating Web page design and the factors that contribute to the kind of performance that leads to repeat business. This chapter reviews some actual case studies and suggests design practices that can improve page performance and perhaps increase a site's capacity for more concurrent visitors.

There is no absolutely correct way to set up a Web site. Multiple design and operational trade-offs exist in setting up even trivial sites. To quantify the affect of trade-offs, IBM uses an internal tool that provides performance data about the site before and after changes. The tool analyzes a page and displays details about the timing, size, identity, and source of each page item; page owners and site operators use these details to identify areas where they can improve performance.

There are five major elements that influence the end user experience:

1. The content of the page (information)
2. The presentation of the content (look and feel)
3. The efficiency of presentation (size)
4. The organization of content (packaging)
5. The administration of the web site (delivery)

Although the page analysis tool cannot measure content and presentation, it can provide information useful in improving efficiency, organization, and delivery. The tool can also be used to help confirm Web page design standards and practices for optimizing performance. Design standards and practices can be prototyped and analyzed to document performance characteristics to be considered along with other considerations (for example, marketing issues regarding presentation appeal). When applied, the tool has revealed areas where significant improvement is possible (and has already been achieved) and suggests page design practices that can help avoid common pitfalls.

Introducing Web communications

The time it takes to download a page results from many factors and the interaction among them. Page designers who best understand these factors can best optimize page download time. A significant factor is how basic and secured Web site communications work. Figure 3-1 is a relatively simplified view of Web communications showing the path of a request from a Web browser to the Web server; note that depending on the configuration of the Web site, there could be multiple servers of more than one type (data, image, proxy) at more than one location.

Overview of Web Communications

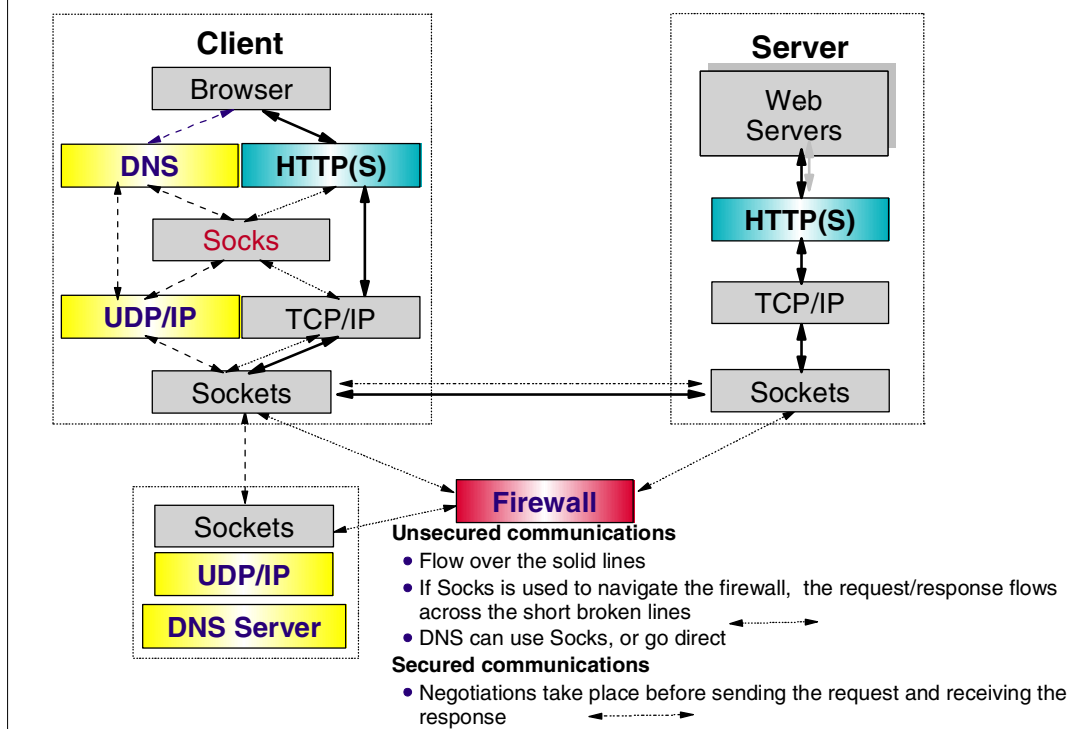


Figure 3-1 Overview of Web communications

A client user enters a request, for example, a Web site name like `www.ibm.com`. The browser, such as Netscape Navigator or Microsoft® Internet Explorer, accepts the request, then typically uses the Domain Name Service (DNS), a service of the User Datagram Protocol (UDP), to resolve fully qualified names (FQNs) like `www.ibm.com` into Internet Protocol (IP) addresses like `123.321.456.34`. DNS builds a connection to the DNS server to obtain the IP address.

When the browser receives the IP address, it initiates a Hypertext Transfer Protocol (HTTP) request. HTTP runs on the Transmission Control Protocol (TCP), which runs on IP, which is the network-layer protocol of the Internet. What happens next depends on whether communications are secured.

If communications are not secured, the browser passes an HTTP request directly through TCP/IP, which creates what is called a socket, a virtual mechanism to manage the addressing needed for sending the request and establishing the connection to the Web server.

If communications are secured, the browser passes a secure HTTP (HTTPS) request to Socks, a security package that negotiates for transmission through the firewall. Such security negotiations occur both before sending the request and before receiving the response. The server also refers to the socket to accept the request and return the response through the firewall.

When the complete end-to-end connection is established, the server fulfills the request by obtaining and serving the items that comprise the page. A page is comprised of one or more text (usually Hypertext Markup Language -- HTML) files, graphic image files (GIFs), audio clips, video clips, and applets. The HTML specification determines the format and content of the page. The operation of sending a file to a client is referred to as a hit on the server. The

time from the browser's request through receipt of the initial reply is called server response time.

The design of the communication protocols used by browsers for Web access creates times when either the Web browser or the Web server must wait for responses from other components. The more time spent in these protocol waits, the longer the delay site visitors may experience while waiting for page content. The 'farther' the browser is from the server, the greater the likelihood of delays due to intermediate links or devices in the path between the browser and the server. A delay could be added by any hardware or software component, including components or subsystems of the browser or the server itself. Even in the best cases, the links and devices in the path act like variable time amplifiers. Each link or device in the path adds a fixed amount of time to perform its function and also has the potential to add significant delay due to queuing related to component saturation. For a discussion of the best practices for scaling Web site capacity, see Chapter 2, "Designing for scalability" on page 11.

When bad things happen to good pages

Many designers of the "first generation" Web sites optimized their design for graphic appeal and relied on the newness of the Web environment to attract visitors and the variety of "eye candy" they could offer to keep visitors there and bring them back. Even today, designers who redesign pages sometimes produce pages that are, from the visitor's viewpoint, prettier but "worse", as in slower, than their predecessor pages. This is no longer acceptable. Web sites that permit customers to transact business must offer their information and services in a way that meets the customers' needs and brings them back for more. That means "performance", usually measured in response time to a customer's request. The goal is to achieve a perfect balance of content and performance.

The major factors that contribute to download time are page size in kilobytes, number and complexity of items, number of servers accessed, and whether SSL is used. Table 3-1 shows the way these factors can be measured. The actual measurements of two related pages demonstrate how measurements between page types can vary, particularly when communications are secured.

Table 3-1 Page download measurements

Measurement	Home	Login (secure page)
Load time (seconds)	6.112	13.59
Size (bytes)	29853	35150
Number of items	8	12
Numbers of servers accessed	1	Unknown
Number of connections	5	6
Failed connections	0	0
Total connection time	5.058 seconds / 31%	3.271 seconds / 9.95%
Average connection time	0.561 seconds	0.27 seconds
Total SSL connection time	Not applicable	4.433 seconds / 13%
Total server response time	7.936 seconds / 48% (little high)	10.999 seconds (SSL) / 13%
Average server response time	0.881 seconds	0.916 seconds (SSL)
Total delivery time	3.206 seconds / 19%	14.174 seconds (SSL) / 43%

Measurement	Home	Login (secure page)
Average delivery time	0.356 seconds	1.181 seconds (SSL)
Address resolution time	0.579 seconds	0 seconds

In Table 3-1, the shaded measurements are all considered good. In our experience to date, we've concluded the following are "good" measurements for a dial-up modem connection as shown in Table 3-2:

Table 3-2 Good measurements

Average server response time	< 0.5 seconds
Number of items per page	< 20
Page load time	< 30 seconds
Page size in bytes	< 64K

While acceptable, these measurements are not considered world class. As an example, the page load time is acceptable at less than 30 seconds, yet we consider load time at less than 20 seconds to be world class. A more specific way to rank load times is shown in Table 3-3:

Table 3-3 Page load time rankings

Seconds to load	Ranking
Less than 10	Excellent
10-15	Very Good
15-20	Good
20-25	Adequate
25-30	Slow
Over 30	Unacceptable

IBM's page analysis tool is designed to explain some of the mysteries about how Web pages are delivered to Web browsers, and to help designers and site operators improve performance and user satisfaction. It does this by revealing details about the timing, size, identity, and source of each item that makes up a page. The details revealed can be used to identify areas where performance improvement could enhance the end user experience.

Figure 3-2 is an example of how the tool presents page download measurements; we added the text boxes for this article to help you understand and interpret the output. Easily installed on and run from the client side, the tool presents the total download time for the page (purple line) and the timing, size, identity, and source for each page item. The tool uses colors to represent different measurements.

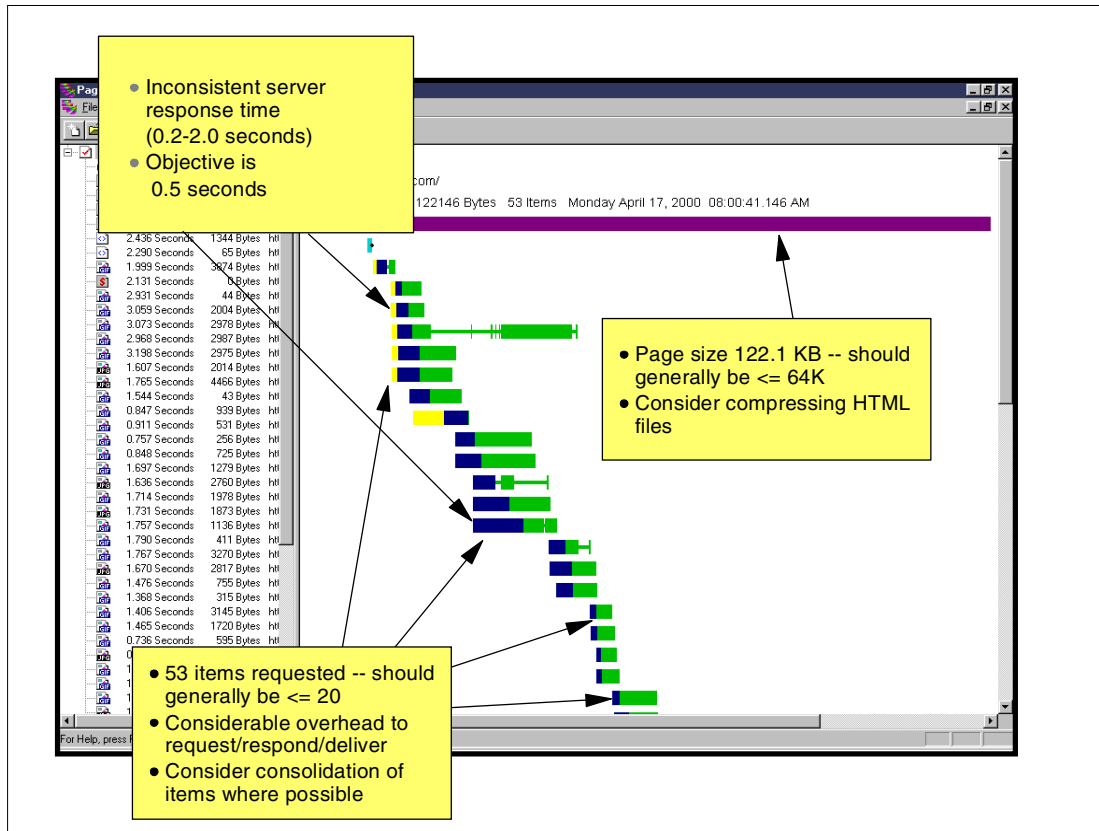


Figure 3-2 Sample measurement of a Web page

Designing, tuning, and redesigning Web pages is fraught with unintended consequences. Certainly designers do not design their pages to load slowly. Here are examples of Web pages whose download time was improved significantly after analysis by IBM's team and tool:

- ▶ For the home page of an online retailer, IBM's tool revealed that the browser was requesting one page item, a spacer GIF, multiple times in a caching environment. Spacer GIFs are commonly used to create white space on the page. Upon analysis, we determined that the HTML references for this item were so close to each other that the response to the first request was not received in the cache before the browser issued another request. By issuing the first request earlier, the response can be cached, thereby eliminating the need for subsequent requests.
- ▶ We've seen 'no cache' set on items in an environment where the browser is capable of caching, while the page at the server was set up to not allow the browser to cache that content, even static GIFs. Failure to use caching for the simple items, such as GIFs, uses up valuable time.
- ▶ Another large online retailer's home page exhibited the common problem of too many items (over eighty). There is no rule on determining the correct size and number of items. The minimum number and size that relays the correct information and loads the fastest is optimal. Oversimplifying, we can take the average overhead of 640 bytes per item. This amounts to 51,200 bytes of overhead to move the 82,592 bytes of graphics data or 62% overhead (51,200/82,592). The time taken to move the image data was 27.83 seconds (we measured and subtracted from load time, the time from the beginning of request to the startup of the first connection that began retrieving an image and subtracted this from the page load time 30.5 - 2.67 = 27.83) or a rate of 4807 bytes/second (51,200 + 82,592 = 133,792 / 27.83 = 4807) for this load.

If we tried to find the optimal load, it would be the four concurrent connections used by the browser sending data constantly with the least amount of overhead. This would be four files (each with a size of $\frac{1}{4}$ the 82,592 bytes) at 20,648 bytes each. The overhead for the four files would be approximately 2560 bytes (4×640) or 3%. It would take approximately 17.71 seconds ($2,560 + 82,592 = 85,152 / 4807 = 17.71$) to load the four files. This says that reducing the number of images and keeping the same amount of data, the maximum time that could be saved is 10.12 seconds ($27.83 - 17.71 = 10.12$).

If the reduction of the number and size of images (for example, by adjusting the number of colors, or the size of the composite image) could be done, then the time savings could be greater. While it's probably not feasible to go from eighty images to four, reducing the number of images makes a significant difference. The only way to optimize this area is with design, prototyping, and measurements to achieve a balance between good presentation and optimal delivery.

- ▶ A large bank implementing online banking was able to reduce the time it took to download its home page from 15 minutes to 26 seconds. Our analysis demonstrated that their download time was slowed by too many items (60), too much data (>115K), and, especially, their unnecessary use of encryption. Eliminating encryption on the home page, upgrading specific hardware components, and some HTTP tuning yielded not only significantly reduced download time but also increased Web site capacity.

What's a good page?

Generally speaking, we observe that pages that load the fastest:

- ▶ Present a few simple, small items, selected for their business value
- ▶ Retrieve items from a single server
- ▶ Combine requests for multiple items from the same server
- ▶ Use persistent connections
- ▶ Request items early
- ▶ Store and retrieve items used more than once from the browser cache
- ▶ Assign private information to private pages and secure private data only
- ▶ Use preproduction utilities that remove extra white space from the source HTML

Pages with features that enable visitors to keep moving appear to load fast, which, from a visitor's standpoint, can be nearly as valuable as actually loading fast. Pages with these features:

- ▶ Present early the links to the site's major sections
- ▶ Use direct links
- ▶ Label visual components that are hot

In the next section, we review how these features translate into page design practices and we discuss the related constraints and trade-offs.

Design practices that can improve performance

IBM's work to-date with several customers with high-volume Web sites suggests there are many practices that when followed reduce the time it takes to download a Web page.

A common theme among many recommended practices is moderation in all things. For example, any page is likely to have multiple items and require multiple connections. What's important is that page designers consider every item first for its business value and second for its size and complexity. Page designers can control for size and complexity and must assure the item's business value, size, and complexity justify the time each contributes to the overall download time. Page designers can also influence the number and types of connections and must understand how the choices they make affect download time.

For efficiency, Web designers and developers tend to locate themselves in proximity to the Web server they are working with. Most try to be on the same LAN. Web site visitors tend to be farther away and may be connected via dial-up at considerably slower speeds. From their viewpoint, the Web designers may not see much difference in response time, but the site visitor will see the benefits of thoughtful packaging. It is a good Web site development policy that developers regularly view pages under development by using connections that are typical for the target users.

Web pages have common components and characteristics that can and should be managed with an eye toward minimizing download time. Doing the "right" thing will not always be possible, and some components or characteristics may be outside of the control of the page designer. Still, everyone with an interest in the site's performance should understand these factors and their related trade-offs:

- ▶ Number, size, and complexity of items
- ▶ Number of connections
- ▶ Number of servers accessed
- ▶ Use of white space
- ▶ Load sequences
- ▶ Data security

Manage number, size, and complexity of items

The number, size, and complexity of page items is the single most significant contributor to page size, page complexity, and the time it takes to download the page. Quite simply, pages with a few, simple items -- selected for their business value -- load the fastest and yield the most satisfied visitors

Number of items: It's impossible to generalize about the correct number of items. After selecting the required items, there are techniques that can help minimize download time:

- ▶ Send a menu as a browser or client-side map instead of a table with individual graphic elements. Tables are inherently slower, especially those with graphic elements.
- ▶ Combine items. The Web server will require fewer machine cycles to retrieve and deliver content.
- ▶ Avoid rollover GIFs. Using mouse rollovers that dynamically change the displayed GIF looks interesting, but requires additional GIFs be downloaded for the effect to operate differently over each menu item. Eliminating the rollover GIF can reduce the number of items.

There are additional techniques, although most trade off some amount of interface function for a reduction in the number of items.

Size: Consider each item's size in relation to the information it is intended to convey or the function it is intended to perform. The larger an item is, the longer it takes to load. Usually, large items are not necessary to deliver intended information or function.

Complexity: The complexity of a page affects how quickly it can be presented. Consider the delays involved when choosing items with features that add complexity. Factors that contribute to page complexity include large tables, table cells whose sizes are dynamically calculated, Java scripts, and Java applets. Animated GIFs, image color management, and image dithering can also contribute delays. These delays vary from browser to browser, and from level to level within a browser; thankfully they tend to become faster with new levels, but not always.

Items formed or described poorly or incompletely can suspend the browser to socket communication. Some tables may be so complex that the browser is fully occupied by their operation and cannot service its socket connections. Time goes on, but nothing in-progress ends. The offending item is probably already at the browser and is being acted on when the hang occurs. Servers and networks can hang also, but a browser hang is almost always reproducible. Such hangs can lead to lost connections, requiring resources to reconnect, adding to overall load time and dissatisfaction of visitors.

The number and size of HTML files are indicators of page complexity. While HTML coding is outside the scope of the HVWS team's analysis, we do know there are utility programs, such as GZIP, available to compress HTML files. We have seen GZIP reduce the size of a HTML file by 80-90%. A smaller HTML file reduces download time and permits the browser to start presenting the page sooner.

Manage number of connections

Information from a Web server reaches the Web browser by way of TCP/IP socket connections. The connection(s) must be opened on both ends before page information can flow. Each connection takes time to set up and take down, and some connections inherently take more time than others. Consider the following for each required connection:

- ▶ Persistent connections can reduce connection setup overhead if multiple items must be transmitted
- ▶ Secured connections take more time to set up

A Web site can have some control over whether it leaves open or closes a socket connection after delivering an item. If the Web site closes connections, the browser must establish a new connection for each item. This type of connection overhead can significantly extend the delay visitors experience when loading pages. Most browsers attempt to keep the connection on their end open, but both ends have to agree that the connection can be held open. The choice to keep a connection open is usually made at the Web server by way of a server configuration option to determine whether the server will support the use of persistent connections when the browser is capable of persistent connections.

High-volume Web sites may wish not to maintain persistent connections because such connections can lead to consumption of all available ports or other constrained server resources like "threads". Additional resources may have to be made available at the server to support persistent connections.

A reasonable objective is to limit the connections per page to four. As servers and HTTP server software evolve, they expand their limits where resources become constrained. Site visitors may benefit when the connections at the server end are kept open.

Manage number of servers accessed

In the best of worlds, the few, simple items would reside on the same server, yielding the fastest possible download time. In the real world, however, page items often reside on more than one server. These items may be from servers at one site or servers across multiple sites. Each time another server is used, the browser must connect a socket to the new server. If all of the browser's connections are being held open, an existing connection must be broken in order to connect to the new server. Many times additional items are required from the first server and the connection must then be reestablished. When possible, organize items to come from a single server. When you must use multiple servers, combine requests from the same server to take advantage of open connections.

Banner advertising is an example of an item that usually resides on a site different from the base page HTML information. When including a banner, specify the banner's dimensions in the base HTML so the browser does not have to calculate the size. Some browsers will not display any content beyond the banner until it has retrieved the banner and calculated its size. Other browsers may start to display the page and then flicker as the size of arriving ad images is calculated and the page is reformatted.

Web sprayers may be in use in front of a Web server's address. A Web sprayer appears as one address to the browser but actually hides multiple servers providing content. Items with long composite times may be coming from an address different from the base page's HTML. Depending on browser design, such items may hold up displaying the page.

Some sites use a technique that allows multiple names for the same server address and can make the site easier and faster to find. For example, <http://www.xyznewscom> and <http://xyznews.com> refer to the same site. If you use this technique, try to avoid switching to the other server name during the page load. Some sites handle the situation by returning a page stub that refers the Web browser to the other name. This results in making the Web browser use more time to look up the other name for the site and establish a new connection before it can retrieve the page content.

Use direct links whenever possible to avoid the cost of an intermediate page. Redirection is best reserved for pointing browsers to a new set of pages when loaded from an old bookmark.

Using the browser's memory cache can reduce download time for a page with multiple requests for the same item. For example, designers use spacer GIFs to position page items. Rather than consecutive requests for spacer GIFs, it's preferable to request a spacer GIF, then request other items. This allows the server time to serve the GIF into the browser's cache so that the GIF is available for subsequent requests. Retrieving the GIF from the cache is faster than returning to the server for each request.

Manage use of white space

Judicious management of white space can help achieve acceptable download times and may even extend the time before a server needs to be added.

Page designers often use white space to help them visualize the page presentation. The browser doesn't need the extra white space to operate properly. Consider using available utilities to remove the extra white space in the source HTML before placing the page(s) on the production Web server.

Avoid the use of extra white space on pages requiring encryption. While extra white space in clear text can be compressed well across a dial-up line, encrypted white space does not compress well because it is no longer a string of repeated character symbols. After encryption, each block of repeating spaces is usually represented by a unique byte string,

making them less likely to be compressed by the modems. Each extra byte costs something to deliver and provides no improvement for site visitors.

Manage load sequences

Designers can sequence requests for items in such a way that download time is optimized. The objective is to specify the sequence such that concurrent operations allow the page to be loaded smoothly. Request items, especially large items and those required for navigation, early to avoid delay at the end of the load sequence. Ideally, the browser should be able to identify items in time to keep its connections to the server busy.

Understand impact of data security

SSL (privacy) handshakes and encryption can be time consuming for both ends. Because privacy creates a drag on each item, it is even more important to design encrypted pages well. The significance of the practices discussed above is escalated when applied to encrypted pages.

HTML items on an encrypted page do not compress well because the HTML is converted to long sequences of numbers that do not work well with dial-up modem compression schemes. This makes the topic about extra white space even more important on an encrypted page.

Clearly, information that is private must be kept private. The balance here is to assign private information to private pages and public information to public pages, and avoid mixing private and public data. The overhead required for the private information should not be invested in any public information.

It is not just about satisfying customers

Of course, your business success depends on satisfied customers. Achieving healthy rates of repeat business on your Web site is your objective. The design practices suggested herein can help you improve the performance of your Web site, which can surely contribute to customer satisfaction. At a minimum, your page designers should adopt these practices as part of their design task:

- ▶ Manage number, size, and complexity of items
- ▶ Manage number of connections
- ▶ Manage number of servers accessed
- ▶ Manage use of white space
- ▶ Manage load sequences
- ▶ Understand impact of data security

Ideally, you should enhance your design team with specific performance expertise. Thus enhanced, your team should be able to develop a site that satisfies customers, reduces consumption of valuable IT resources, and simplifies your operations. When implemented, these practices may also help you increase the capacity of your site as you serve more concurrent visitors with the same hardware.



Planning for growth

As e-business and its related requirements grow at "Web speed", a critical issue is whether the IT infrastructure supporting the Web sites has what it needs to provide available, scalable, fast, and efficient access to the company's information, products, and services. More than ever, CIOs and their teams struggle with the challenges to minimize downtime and network bottlenecks and maximize the use of the hardware and software that comprises their e-business infrastructure.

The IT infrastructure supporting most high-volume Web sites (HVWSs) typically has multiple layers of machines, frequently called tiers, and each tier handles a particular set of functions, such as serving content (Web presentation servers), providing integration business logic (Web application servers), or processing database transactions (transaction and database servers). Figure 4-1 shows an e-business infrastructure comprised of several tiers. Each tier consists of multiple machines, from two to hundreds, to provide capacity and availability for the functions running at that tier. The IBM WebSphere software platform for e-business includes edge servers, Web application servers, development and deployment tools, and Web applications.

Infrastructure for e-business

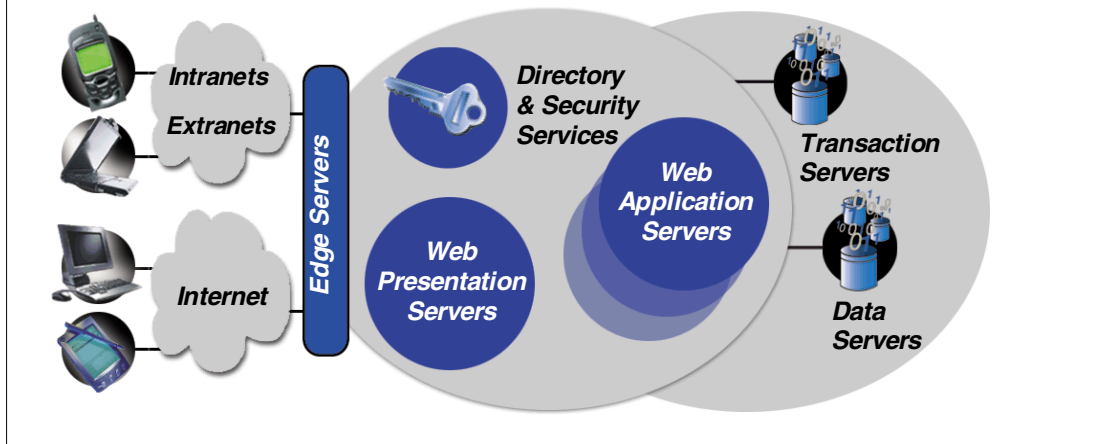


Figure 4-1 Multi-tier infrastructure for e-business

Even with this growing complexity, typical IT infrastructures can be analyzed and related models developed to assist in predicting and planning how to meet future requirements. This chapter presents a methodology for IT professionals to use to determine whether their Web site can satisfy future demands and to evaluate potential workload and infrastructure changes. It also introduces the concept of configuring a Web site based on an analysis of how different components combine to best meet the performance objectives of your particular workload pattern, potentially reducing the costs of prototyping and stress testing.

IBM's IT experts have been working with many IBM customers to analyze large Web sites and help customers implement scalable Web sites. Some of these customers are already working with IBM to exploit and further the technologies for capacity planning being developed. Our methodology is based on this on-going research as well as IBM's patterns for e-business.

Introducing a methodology for capacity planning

Our methodology for capacity planning is based on our analysis of many large Web sites, including IBM's, and continuing engagements with large customers seeking to improve site performance, accurately project workloads, and make infrastructure changes that will satisfy future requirements. The methodology consists of four steps:

1. Identify your workload pattern
2. Measure performance of current site
3. Analyze trends and set performance objectives
4. Model your infrastructure alternatives

The steps are introduced below, then described in more detail throughout the rest of the chapter. These steps are useful whether you are considering changes to a current site or planning a new site. The chapter focuses on technologies for capacity planning. Implementing such technologies will affect your IT organizations, processes, and people; the related implications are not addressed in this chapter.

1. Identify your workload pattern. Your workload is assumed to be high-volume and growing, serving dynamic data, and processing transactions. Beyond that, you must consider other characteristics, such as transaction complexity, data volatility, and security. After your analysis, it becomes clear that your workload pattern fits into one of five classifications: *publish/subscribe*, *online shopping*, *customer self-service*, *trading*, or *business-to-business*. Correctly identifying your workload pattern assures the best results from the remaining steps and maximizes your site's chances for satisfying future requirements.

2. Measure performance of current site. As always, you must understand the present before planning the future. You need to measure these site characteristics: volumes (hits, page views, transactions, searches), arrival rates, response times by class, user session time, number of concurrent users, and processor and disk utilization. If you're planning a new site, you'll need to estimate these metrics; IBM's high-volume Web site team can provide typical site profiles.

3. Analyze trends and set performance objectives. Your workload is growing and your current metrics, no matter how good they are, must improve, along with the capacities of the hardware and software that comprise your infrastructure. In this step, you analyze trends to determine future peak volumes, then set objectives for each metric identified in Step 2, along with any new metric that applies to your future requirements.

4. Model your infrastructure alternatives. At this point, you are ready to determine the components needed to construct your site's infrastructure. IBM can help you match components to the particular requirements and objectives of your workload pattern.

Our methodology for capacity planning complements IBM's patterns for e-business by referring to:

<http://www.ibm.com/developerworks/patterns>

in that each High-Volume Web Sites (HVWS) workload pattern can be mapped to one of the e-business patterns for site design. Regardless of the methodology you used to design your site, our capacity planning methodology can complement that effort and establish a foundation for managing your capacity requirements.

Step 1. Identify your workload pattern

Your workload is assumed to be high-volume and growing, serving dynamic data, and processing transactions. Beyond that, you must consider other characteristics, such as transaction complexity, data volatility, security, and others. After your analysis, it becomes clear that your workload pattern fits into one of five classifications: *publish/subscribe*, *online shopping*, *customer self-service*, *trading* or *business-to-business*. Correctly identifying your workload pattern assures the best results from the remaining steps and maximizes your site's chances for satisfying future requirements.

Review the workload pattern descriptions below to identify your workload pattern. You may also want to refer to Table 1-1 on page 7 in Chapter 1.

Publish/subscribe Web sites provide users with information. Sample publish/subscribe sites include search engines, media sites, such as newspapers and magazines, and event sites, such as those for the Olympics and for the championships at Wimbledon. Site content changes frequently, driving changes to page layouts. While search traffic is low in volume, the number of unique items sought is high resulting in the largest number of page views of all site types. As an example, the Sydney Olympics site successfully handled a peak volume of 1.2 million hits per minute using IBM's WebSphere Application Server, WebSphere Commerce Suite, and MQSeries. Security considerations are minor compared to other site types. Data volatility is low. This site type processes the fewest transactions and has little or no connection to any legacy systems.

Online shopping sites let users browse and buy. Sample sites include typical retail sites where users buy books, clothes, and even cars. Site content can be relatively static, such as a parts catalog, or dynamic where items are frequently added and deleted as, for example, promotions and special discounts come and go. Search traffic is heavier than the publish/subscribe site, though the number of unique items sought is not as large. Data volatility is low. Transaction traffic is moderate to high, and almost always grows. The typical daily volumes for many large retail customers, running on IBM's WebSphere Commerce Suite, range from less than one million hits per day to over 13 million hits per day, and with a range from 100,000 transactions per day to three million transactions per day in the top range; of the total transactions, typically between 1% and 5% are buy transactions. When users buy, security requirements become significant and include privacy, nonrepudiation, integrity, authentication, and regulations. Shopping sites have more connections to legacy systems, such as fulfillment systems, than the publish/subscribe sites, but generally less than the other site types.

Customer self-service sites let users help themselves. Sample sites include banking from home, tracking packages, and making travel arrangements. Data comes largely from legacy applications and often comes from multiple sources, thereby exposing data consistency. Security considerations are significant for home banking and purchasing travel services, less so for other uses. Search traffic is low volume; transaction traffic is low to moderate, but growing.

Trading sites let users buy and sell. Of all site types, trading sites have the most volatile content, the highest transaction volumes (with significant swing), the most complex transactions, and are extremely time sensitive. Products like IBM's WebSphere's Application Server play a key role at these sites. Trading sites are tightly connected to the legacy systems, for example, using IBM's MQSeries for connectivity. Nearly all transactions interact with the back end servers. Security considerations are high, equivalent to online shopping, with an even larger number of secure pages. Search traffic is low volume.

Business-to-business sites let businesses buy from and sell to each other. Data comes largely from legacy applications and often comes from multiple sources, thereby exposing data consistency. Security requirements are equivalent to online shopping. Transaction volume is moderate, but growing; transactions are typically complex, connecting multiple suppliers and distributors. There are two styles of this pattern:

1. *Business-to-business integration*: this style includes programmatic links between arms-length businesses (where a trading partner agreement might be appropriate). Example: supply chain management.
2. *eMarketplace or B2M2B*: the M represents the eMarketplace, which supports multiple buyers and suppliers. The buying function can be performed online or programmatically. Example: e-Marketplace.

Step 2. Measure performance of current site

As always, you must understand the present before planning the future. You need to measure these site characteristics: volumes (hits, page views, transactions, searches), arrival rates, response times by class, user session time, number of concurrent users, and processor and disk utilization. If you're planning a new site, IBM has site profiles you can use to estimate these metrics.

Our analysis of the performance of e-business infrastructures under various workload patterns demonstrates that workload pattern complexities (for example, bursty arrival patterns) can significantly affect resource demands, throughput, and the latency encountered by user requests, in terms of higher average response times and higher response time variance. Without adaptive, optimal management and control of resources, service level agreements (SLAs) based on response time are impossible. The capacity requirements on

the site are increased while its ability to provide acceptable levels of performance and availability diminishes.

When analyzing your current site, do not overlook the design of your Web pages. IBM's work to-date suggests there are many practices that when followed reduce the time it takes to download a Web page. Web pages have common components and characteristics, such as page size and number of items, that can and should be managed with an eye toward minimizing download time. Doing the "right" thing will not always be possible, and some components or characteristics may be outside of the control of the page designer. Still, everyone with an interest in the site's performance should understand these factors and their related trade-offs. Figure 4-2 summarizes page design metrics from 15 different Web sites; the metrics vary but strongly suggest that page design is an important performance component that when managed well can improve a site's capacity. Metrics considered good or excellent are shaded green; less favorable metrics are shaded with yellow, and unacceptable metrics are shaded with red. For more information, see Chapter 3, "Designing pages for performance" on page 27. The IBM WebSphere Studio Page Detailer component is a tool that can help identify the types of information shown in Figure 4-2, but also graphically illustrates how these components can affect page responsiveness.

Example of Web page metrics						
Web page	Page load time (sec)	Page size (bytes)	Number of items	Number of connections	Number of servers	Failed connections
1	32.33	179,968	51	17	2	0
2	30.5	140,842	80	7	2	0
3	31.78	136,943	25	6	1	0
4	26.26	122,146	53	7	1	0
5	78.26	121,664	56	21	3	0
6	41.648	111,281	37	5	2	0
7	34.45	105,433	35	21	2	0
8	22.18	93,580	29	6	1	0
9	22.52	84,240	46	46	1	0
10	27.03	72,411	36	36	4	0
11	19.951	64,347	30	19	1	0
12	29.741	61,073	40	11	1	0
13	15.14	56,430	25	5	1	0
14	15.69	43,891	23	23	1	0
15	8.77	39,189	12	5	2	0

Figure 4-2 Examples of Web page metrics

Understand workload metrics

Correctly identifying your workload pattern prepares you for measuring and understanding the complexities of your site. Each workload pattern has an associated class of user requests. Figure 4-3 shows an example of classes of user requests associated with the online shopping workload pattern.

Each class is characterized by how the requests arrive at the Web site and the resources required to satisfy the request. The major factors affecting arrivals include standard (marginal) distribution, dependence structure and seasonality. In general, IBM's analysis demonstrates complex behaviors that include light-tailed and heavy-tailed distributions, short-range and long-range dependencies, strong seasonality and periodicity, and geographic effects. The typical assumption of independent exponential interarrivals of Web requests does not hold true under these conditions and one has to solve the problem with nontraditional assumptions that require complex mathematical algorithms. IBM's mathematical research of these

characteristics has enabled the development of improved models to understand and predict the impacts of these relationships and behaviors.

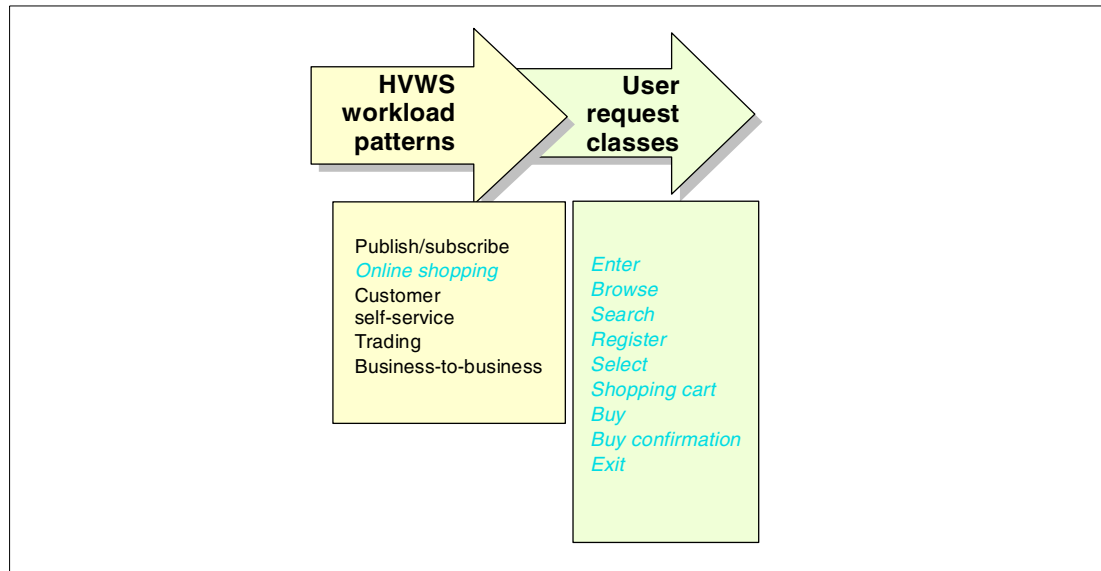


Figure 4-3 Each workload pattern has an associated class of user requests

Distribution and dependence Web traffic exhibits bursty, heavy-tailed, and correlated arrival patterns. Bursts refer to the random arrival of requests, with peak rates far exceeding the average rates. These bursts are caused by unpredictable events such as major stock market swings or special events such as Christmas or Valentine's Day. Such events yield dependencies among requests (for example, larger bursts tend to occur in close proximity), heavy-tail distributions (for example, very high variability in the sizes of the bursts), and the combination of dependencies and heavy-tail distributions. A heavy-tailed distribution for a random variable is one where the tail of the distribution decreases sub exponentially. For these distributions, the probability that a large value occurs is nonnegligible. The batch arrival process exhibits such heavy-tailed behavior and the batch request sizes tend to be strongly correlated. A practical consequence of burstiness and heavy-tailed and correlated arrivals is difficulty in capacity planning.

Burstiness and wide-ranging hit rates are among the most obvious workload pattern complexities that affect Web site performance and availability. In traditional models, requests are independent and the variance in the burst sizes are relatively small. These distributions belong to the class of light-tailed distributions. The burstiness of a HVWS yields heavy-tailed distributions and a strong dependence structure. This is illustrated by the traffic patterns from the 1998 Nagano Olympic games, as shown in Figure 4-4. Such bursts of requests are triggered by some special event, for example, in this case when Japan won the Gold medal for ski jumping in Nagano. Contrast the heavy-tailed distribution from Asia with the light-tailed distribution on the same day from Europe. Further, note the dependence structure from day to day, as well as within each day, at both locations.

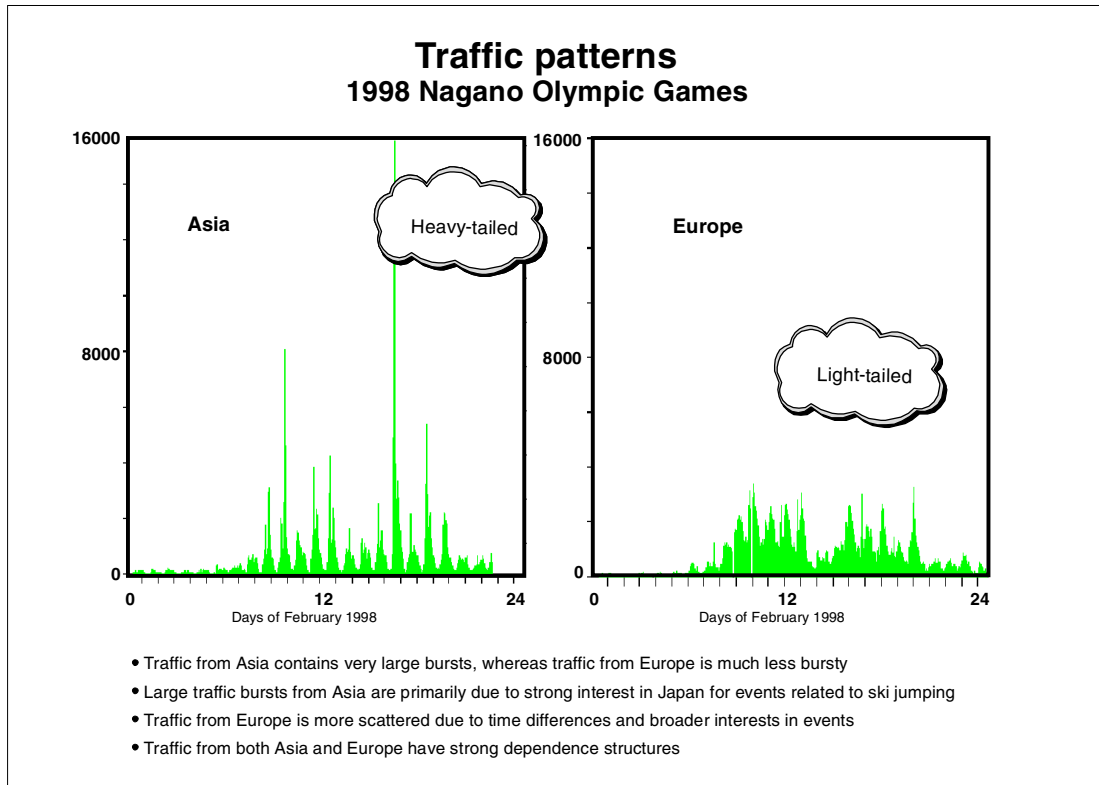


Figure 4-4 Traffic patterns from Nagano Olympic Games

IBM's Wimbledon 2000 Web site also exhibited extreme bursts on its busiest day, 7 July 2000. Figure 4-5 graphs the record-breaking site traffic on that day when peak hits per minute reached 963,948 and peak hits per day totalled 281,605,872).

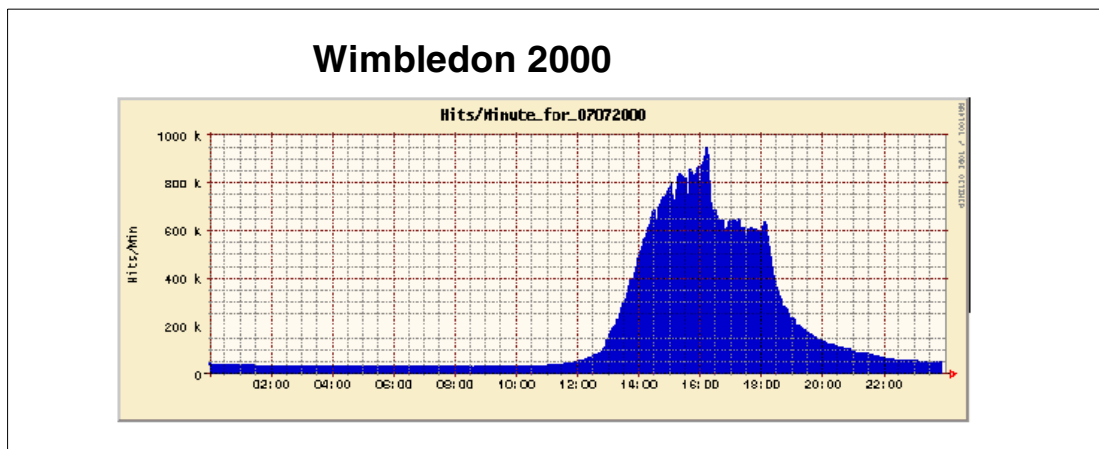


Figure 4-5 IBM's Wimbledon Web site on its record-breaking day

The foregoing nontraditional request traffic stresses the Web server. Bursty traffic with heavy-tailed distributions degrades the performance by several orders of magnitude over light-tailed distributions. For heavy-tailed distributions, the extremely large bursts occur relatively more frequently than the light-tailed model. Moreover, the dependence structure causes these bursts to occur in close proximity to each other. With such input traffic characteristics, the performance measures, in particular, the response time, have similar characteristics as the input traffic. This helps to explain why some sports and e-business Web

sites are more difficult to maintain than relatively simple Web sites (for example, a university Web site serving only static content).

With respect to SLAs, a more powerful set of servers is needed to achieve the same level of service for heavy-tailed distributions in comparison with the case of independent light-tailed request traffic. To guarantee good performance, we need to focus on peak traffic duration because it is the huge bursts of requests that most degrade performance. That is why some busy sites require more "head room" (spare capacity) to handle the volumes; for example, a high-volume online trading site reserves spare capacity with a ratio of three to one.

Seasonality: Seasonality refers to the periodicity of the request patterns. Seasonal traffic is most often represented by the regular daily activities of the users of a Web site. For example, traffic to some e-trading Web sites has consistent peaks and valleys each day when the market opens and closes. Seasonal traffic is also observed in monthly intervals, for example, when users pay bills at the end of the month, and during designated periods, for example, the holiday season.

Figure 4-6 shows an example of seasonal traffic from the Nagano Olympic Web site. The figure plots the number of requests received every five minutes by all servers from Monday, February 9th through Sunday, February 16th. While each daily cycle varies considerably, note that each day has three peaks and that overall traffic intensity increases each weekday then decreases on the weekend. These patterns repeated each week, demonstrating seasonal variations that correspond to weekly cycles.

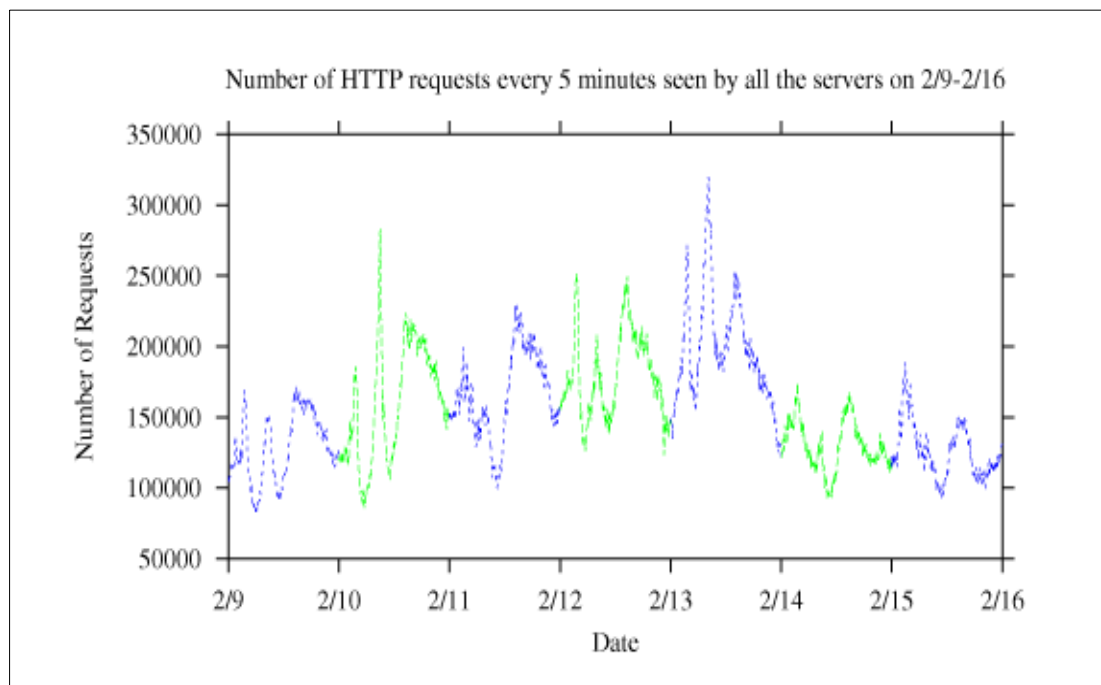


Figure 4-6 Example of seasonality demonstrated by one week from Nagano

Seasonal requests can degrade the performance of the Web server because, for the peak duration, large batches of requests occur around the same time. The central questions are how high is the peak and how long is the peak duration. The answers to these two questions can have a significant impact on how powerful the Web server should be in order to handle a specific SLA. To satisfactorily handle request traffic, the capacity of the Web server should be close to peak request level, with some "head room" to allow for unexpected growth.

Other factors that define the workload pattern include the volume of page views and transactions, the volume and type of searches, the complexity of transactions, the volatility of the data, and security considerations.

The rest of this section introduces techniques available to obtain the measurements you need to complete your capacity plan.

Obtain site measurements

Each workload pattern requires specific measurements. Table 4-1 is an example of some current measurements of an online shopping site.

Table 4-1 Online shopping site measurements

Measurement	Today
Concurrent users	40,000
Hits/second	3,480
Response time in seconds	28
Pages/second	346
Pages /visit	10.6
Visits/second	32.6
Minutes/visit/user	20
Ratio of user visit type	93% browse only 6% browse/search 2% buy

By analyzing typical user visits, it's possible to create probabilities about future user visits. Online shoppers, for example, typically browse, may search, and occasionally buy. You can develop various scripts to describe user visits. Tables 4-2, 4-3, and 4-4 contain samples of scripts for online shopping, online banking, and online trading.

Table 4-2 Online shopping script

Browse	Home page Choose department (static HTML) Choose category Choose subcategory Choose product 1 Choose product 2 Choose department (dynamic category display) Choose category Choose subcategory Choose product 1 Choose product 2
Search	Home page Select product search Submit keyword Select new search Submit keyword

Buy	Home page Select "AtHome" department Select "Candles" category Select "Scented" subcategory Select "tripod candle" Select "Add to shopping bag" Select "Checkout" Select "Complete order online" Select "Charge it"
-----	---

Table 4-3 Online banking script

	Login Force PIN change Main menu Add a payee Schedule 6 bill multi-payment Edit a payment Customize Financial summary Account details Request a check copy Verify check copy request Sign-off
--	--

Table 4-4 Online trading script

	Login Query position Get quote 1 Get quote 2 Get quote 3 Get quote 4 Get quote 5 Trade - buy Check status Get quote 6 Get quote 7 Get quote 8 Trade - Sell Check status Logoff
--	--

Using the scripts and the data from your measurements, you can create what is called a transition matrix. Figure 4-7 is an example of a transition matrix for an online shopping visit. Viewing the sample transition matrix as it relates to the sample script above, you can easily see the browse and search requests; the buy request occurs when the user decides to add (to shopping bag) and pay.

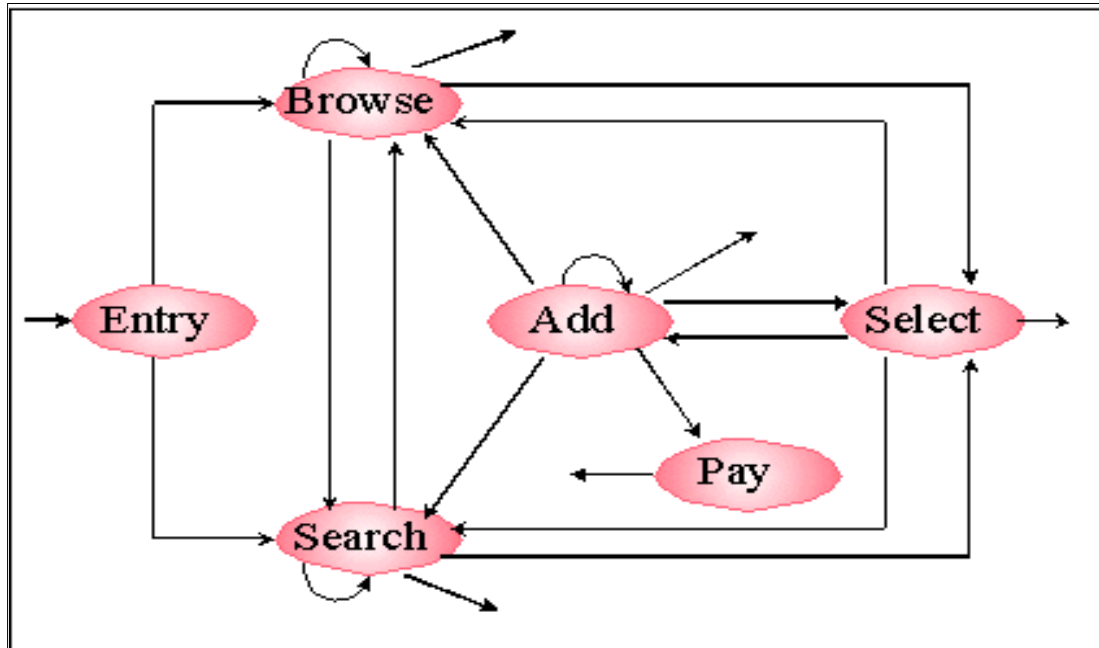


Figure 4-7 Example of a transition matrix for an online shopping visit

Step 3. Analyze trends and set performance objectives

Your workload is growing and your current metrics, no matter how good they are, must improve, along with the capacities of the hardware and software that comprise your infrastructure. In this step, you analyze trends to determine the characteristics of future peak volumes, then set objectives for each metric identified in Step 2, along with any new metric that applies to your future requirements. Table 4-5 is an example of current and projected measurements for an online shopping site. Performance objectives are usually driven by business objectives, for example, to improve response time to preferred customers.

Table 4-5 Projected measurements for online shopping site

Measurement	Today	Projected
Concurrent users	40,000	100,000
Hits/second	3,480	8,700
Response time in seconds	28	< 10
Pages/second	346	865
Pages /visit	10.6	10.6
Visits/second	32.6	81.6
Minutes/visit/user	20	20
Ratio of user visit type	93% browse only 6% browse/search 2% buy	92% browse only 6% browse/search 2% buy

The ability to accurately forecast request patterns is an important requirement of capacity planning. Our forecasting methodology is based in part on constructing a set of mathematical methods and models that isolate and characterize the trends, interdependencies, seasonal effects, randomness, and other key behavior in the Web site's request patterns. This includes, for example, the use of piece wise autoregressive moving

average (ARIMA) processes combined with heavy-tailed distributions. Figure 4-8 is an example of a request pattern we would consider for our models; it shows the number of requests received per second over the period of one hour at the Nagano Olympic site. The curve fitted to these measurements is shown in the middle of the data points.

Since each of the key traffic characteristics can scale differently, we use a general set of mathematical methods to estimate the intensity of each characteristic in future request patterns and to scale each characteristic to its forecasted intensity. We then combine these scaled mathematical models to characterize and predict request patterns. By using our mathematical methods to isolate, characterize, and forecast the trends, interdependencies, seasonal effects, randomness, and other key behavior in the request patterns, we have developed a general methodology that provides a more accurate and effective approach for predicting request patterns than other approaches being used today.

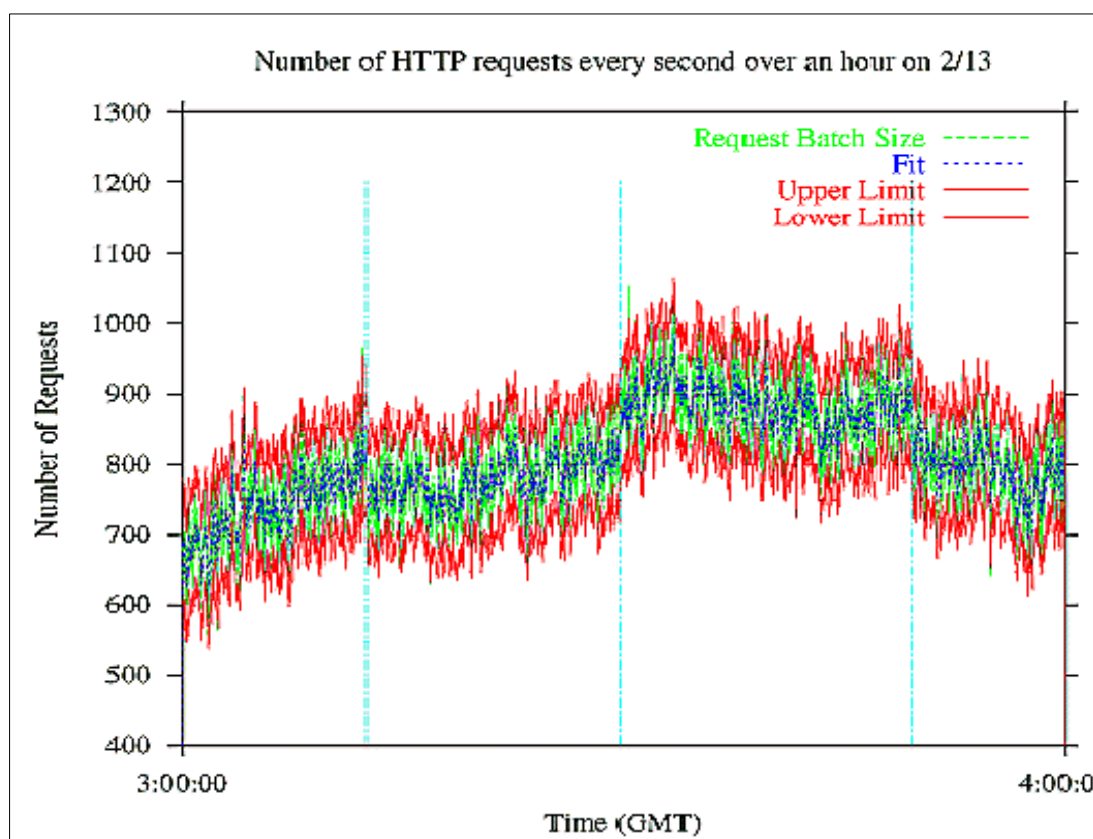


Figure 4-8 Requests per second over one hour during Nagano Olympics

Our methodology has proven effective in practice, having been used to predict request patterns of actual sites over both short-term and long-term time frames. In particular, we used our methodology to predict the peak hour request volumes for a recent sporting event Web site hosted by IBM based on request patterns from the Web site in three previous years, as shown in Figure 4-9, together with 95% confidence intervals. The arrows illustrate one of our approaches, namely a first-order rate-of-change method, for estimating the traffic intensity scaling factor from year to year. Note the exponential growth seen as you go from 1997 to 2000. We then used the forecasted scaling factor for 2000 and our methodology to scale the peak hour traffic model from 1999 to obtain our forecasted peak hour traffic model for 2000. Our forecasts were found to be in excellent agreement with the site's actual peak volumes. Moreover, these methods have been applied with equal success to estimate request patterns for upcoming seasonal events, such as the Christmas online shopping rush. We also used our methodology to forecast request patterns over short-term time frames (days, weeks);

again, our forecasts were found to be in excellent agreement with the site's actual request patterns.

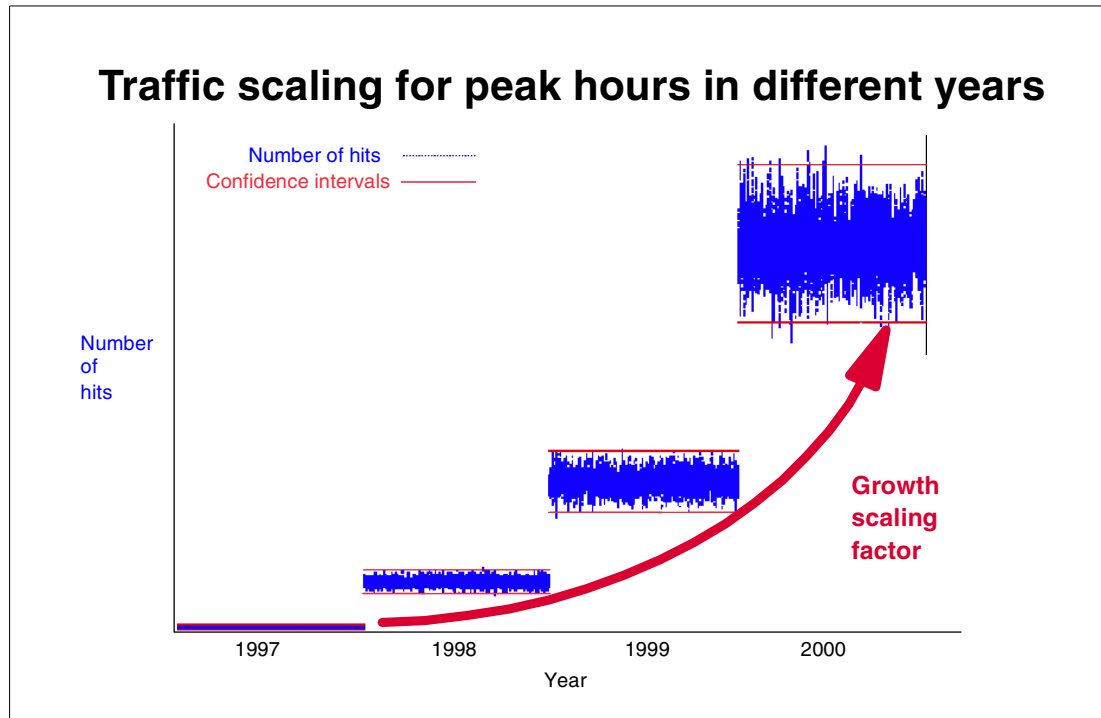


Figure 4-9 Traffic scaling for peak hours in different years

Step 4. Model your infrastructure alternatives

At this point, you are ready to determine the components needed to construct your site's infrastructure. IBM can help you match components to the particular requirements and objectives of your workload pattern.

The technologies IBM is developing for HVWS capacity planning rely on the three models depicted in Figure 4-10.

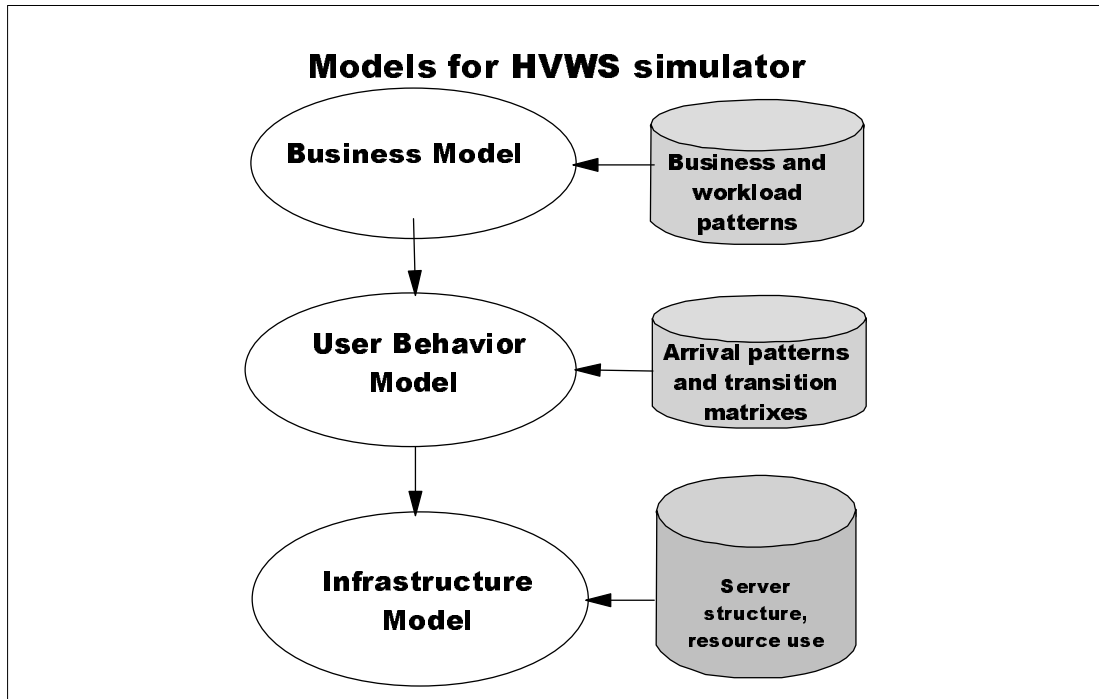


Figure 4-10 Models for HVWS capacity planning

The business model, or business usage model, defines the e-business pattern and workload pattern. There is a user behavior model for each workload pattern. Each workload pattern consists of several classes of user requests. The arrival patterns and routes (transition matrixes) site visitors follow for each class comprise the user behavior model. The hardware and software resources, and the amount of each required to satisfy each class of user requests, comprise the infrastructure model.

The infrastructure model processes browse, search, and buy transactions. The model assumes:

- ▶ The Web site has multiple layers of machines, or tiers, each handling a particular set of functions, such as the site depicted in Figure 4-11 (the figure does not include firewall layers).
- ▶ A load-balancer, such as the network dispatcher, routes requests to multiple Web front-end servers using an algorithm to distribute requests evenly among the servers.
- ▶ The front-end Web servers handle requests for static or dynamic pages.
- ▶ The Web application server processes business logic for the transactions initiated by the request. In Figure 4-11, the account and quote servers are the application servers.
- ▶ The back-end database server handles requests for dynamic pages that involve obtaining or updating information; such requests do not return through the load balancer.

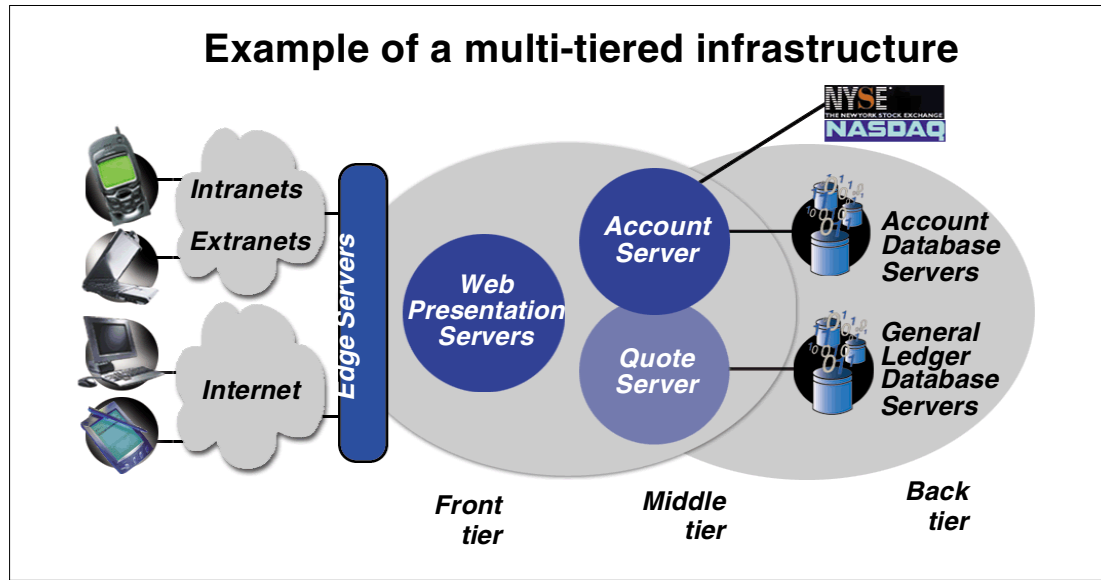


Figure 4-11 A Web site with multiple tiers

We formulate a class of queuing networks to model multi-tier architectures in order to analyze performance at different levels. We further derive a variety of solutions to these models under different input traffic patterns and at different time-scales. This family of mathematical models and solutions are general enough to abstract the underlying hardware and software details, but detailed enough to produce meaningful performance results. We consider the queuing system, where the resources of each tier is modeled by multiserver queues that have specific relationships to one another. These relationships are determined through measured or estimated workload characteristics. We then solve the performance/capacity problem against a set of user requirements, such as the number of concurrent users, response time, or throughput rate. We have also developed unique formulas to allow us to estimate the behavior of the system where peak demands are significantly higher than average demand, and there is a nonnegligible probability of accessing large amounts of data by the user. Our method is flexible enough to model horizontal and vertical scaling, or a combination, depending on user requirements and workload characteristics. For example, we can increase the number of Web servers by adding another server or by adding another processor on a given server. Given the appropriate Web site and workload data, we are then able to obtain performance estimates from our performance models and analysis.

We have developed capacity plans for a number of IBM-hosted Web sites. Figure 4-12 depicts one such site and reflects the process of calibrating our model using current data from the site, then developing projections based on current data, trends, and objectives. In Figure 4-12, the first three response time curves reflect the current data used to calibrate the model as discussed in Step 2. By analyzing current metrics and component information, we are able to project the fourth curve.

Referring to Figure 4-12, the results show that when the request traffic is light, one front-end server is enough to handle the load. As the traffic increases, the response time curve remains flat until the front-end CPU reaches a utilization of 90% (2.8 million hits/hour). At this point, a minor increase in the load can rapidly plunge the system into a situation resembling deadlock, where the front-end server attempts to serve more and more files at slower and slower speeds, such that few are experiencing satisfactory response times. This means that the front-end server has become the bottleneck. We therefore need to add a front-end server, and upon doing so, the front-end CPU utilization drops as desired. The response time curve becomes flat until the front-end CPU again reaches a utilization of 90% (5.1 million hits/hour),

when we need to consider adding another front-end server. The back-end server becomes the bottleneck only after about 15 front-end servers are handling around 28 million hits/hour.

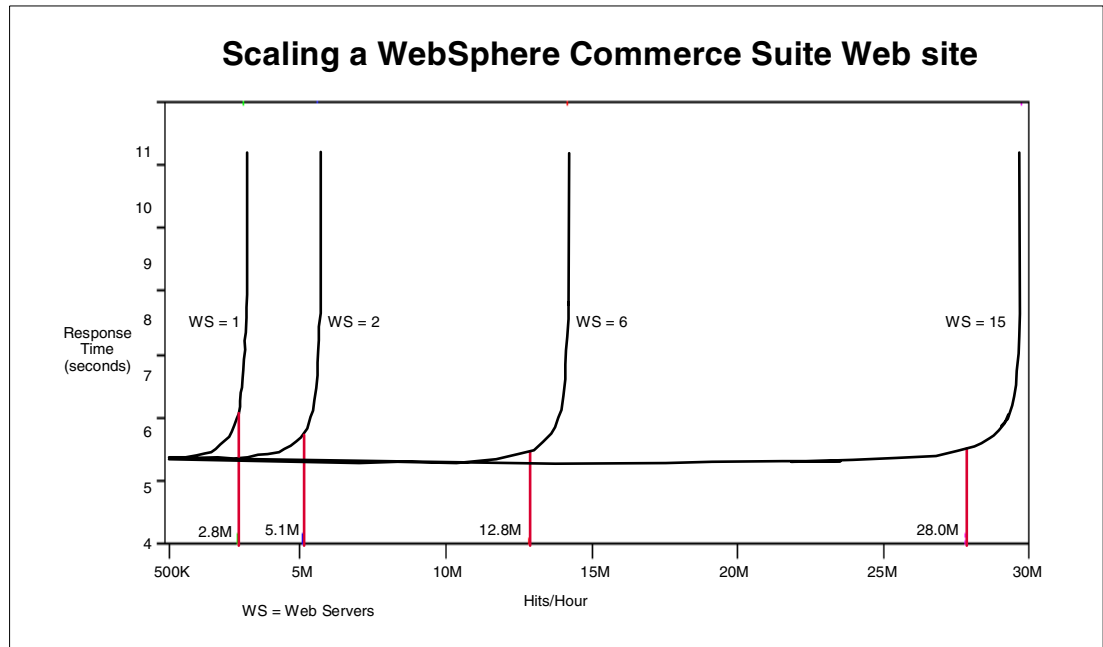
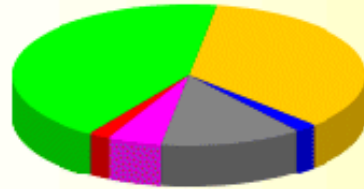


Figure 4-12 *Scaling a WebSphere Commerce Web site*

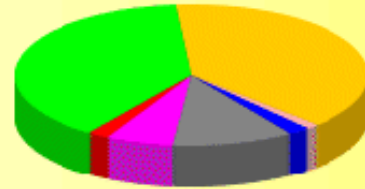
Figure 4-13 is a sample of a graph we produce when analyzing performance objectives against specific hardware and software components.

Components of performance

Lower bound calculations: Upper bound calculations:



Tier 1 CPU service	35%
Tier 1 CPU wait	43%
Tier 1 disk service	2%
Tier 1 disk wait	0%
Tier 2 CPU service	5%
Tier 2 CPU wait	13%
Tier 2 disk service	2%
Tier 2 disk wait	0%



Tier 1 CPU service	39%
Tier 1 CPU wait	39%
Tier 1 disk service	2%
Tier 1 disk wait	0%
Tier 2 CPU service	6%
Tier 2 CPU wait	11%
Tier 2 disk service	2%
Tier 2 disk wait	1%

Figure 4-13 Sample graph showing components of performance

Summary

The challenge of effective capacity planning for high-volume Web sites is an awesome one, but not insurmountable. The methodology suggested offers a road map toward understanding your workload pattern and current metrics, analyzing trends and setting objectives for the future, and, finally, selecting the IT infrastructure components needed to meet your performance objectives. The ability to analyze your site's requirements in the context of your particular workload pattern can contribute greatly to making the right selections. IBM's HVWS team is available to assist you in the application of these steps and the development of appropriate models for your environment. And IBM's WebSphere and MQSeries products, as evidenced by their extraordinary performance at the Sydney Olympics, are significant parts of IBM's solution for rapidly growing e-business infrastructures.

Of course, the subject of capacity planning is an ongoing study. Increasingly valuable information is available, as well as exciting new offerings from IBM, such as the Capacity Advantage tool and "capacity on demand" options that will greatly enhance the ability to respond to traffic growth. With an eye toward discovering and documenting modern design practices that allow ever greater capacities and scalability, IBM's HVWS group is refining its methodology, developing tools that embody that methodology, fine tuning its mathematical methods, and looking at the additional challenges presented by such areas as network caching and the fast-growing business-to-business workloads. The promise remains great, however, of meeting future needs while planning effectively and reducing costs.

References

See the following at the High Volume Web Sites Web page:

<http://www.ibm.com/websphere/developer/zones/hvws>

- *IBM High-Volume Web Sites, Design for Scalability*, December 1999
- *IBM High-Volume Web Sites, Web Site Personalization*, February 2000
- *IBM High-Volume Web Sites, Design Pages for Performance*, May 2000

A.K. Iyengar, M.S. Squillante, L. Zhang. *Analysis and characterization of large-scale web server access patterns and performance*. World Wide Web, Vol. 2, 1999.

W.N. Mills, M.S. Squillante, C.H. Xia, Li. Zhang. *Web workload service requirement analysis: A queueing network approach*. IBM Research Report, 2000.

Z. Liu, N. Niclausse, C. Jalpa-Villanueva. "Web traffic modeling and performance comparison between HTTP1.0 and HTTP1.1", *System Performance Evaluation: Methodologies and implications*, pp. 177-189, 2000.

M.S. Squillante, D.D. Yao, L. Zhang. "Internet traffic: Periodicity, tail behavior and performance implications", *System Performance Evaluation: Methodologies and Applications*, pp. 23-37, 2000.

M.S. Squillante, D.D. Yao, L. Zhang. *Web traffic modeling and web server performance analysis*. IEEE Conference on Decision and Control, December 1999.

D. A. Menasce and V. A. F. Almeida. *Capacity Planning for Web Performance: Metrics, Models, and Methods*, Prentice Hall, 1998.

D.A. Menace and V.A.F. Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*, Prentice Hall, 2000.



Maximizing Web site availability

What happens when your Web site is down? At the least, your site's users become idle, your staff goes into crisis mode, and everyone's productivity decreases. At the worst, the users are your customers and they go somewhere else to buy the product or service your company offers. Depending on how long the site is down, customer satisfaction and revenue are diminished, and the reputation of your IT organization, if not your company, is damaged. It's not a good place to be. That's why availability is such an important aspect of managing a successful e-business Web site.

This chapter reviews availability concepts, all of which are well known to IT professionals. Given the increasing complexities of managing e-business infrastructures and the increasing intolerance for any outages, the chapter stresses that the basic concepts and practices of availability are more important than ever. It also reviews practices specifically related to e-business infrastructures; understanding and applying these practices appropriately can improve your availability statistics and help you achieve your target of continuous availability.

Introduction

IT professionals have been designing systems for high availability for a long time. Most of these systems were implemented in highly structured and controlled environments. With the advent of the Internet and e-business, you now have potentially millions of users, all around the world, operating at all times, and using client/server connections that are not under the control of your IT organization. We are in an era when IT and information do more than simply support a business; they provide businesses a competitive edge in the marketplace. Small outages, tolerated just a few years ago, can now mean significant losses of revenue, which can be quantified, as well as lost opportunities for future business. e-business, globalization, industry consolidation, merger mania, and server consolidation are the forces setting new standards for availability.

Today's high volume e-business Web sites require the highest possible availability to ensure they achieve their business goals for customer satisfaction, repeat business, and profitability. Large e-business Web sites consist of several tiers of servers, each of which have unique availability characteristics and represent possible points of failure. This increases the challenge IT faces in providing a continuously available Web site. Highly available systems are necessary to support corporate strategies and provide a competitive advantage. Increasingly, e-business sites are demanding continuous availability. No time for planned outages and certainly no time for unplanned outages is fast becoming the norm. The luxury of having a window of time to perform system backups has vanished, along with the idea that one can risk the possibility of an unplanned outage. In the increasingly seamless world of e-business, customer relationship management, and supply chain management, where financial transactions depend on a chain of events, no link in the availability chain can be broken and certainly, the central engine of commerce, the server, cannot be down. The corporate strategy demands 7/24 availability of service and IT must support this corporate strategy.

Figure 5-2 shows that a focus on availability is appropriate during all phases. This chapter reviews availability concepts and techniques pertinent to high volume Web sites. IBM recognizes that the complexities of e-business infrastructures make the basics of managing availability more important than ever.

Availability concepts and costs

This section discusses availability, high availability, and continuous availability.

Availability > high availability > continuous availability

In IT, a system, application, or component that can be used is considered to be available. Availability is the measurement of the time the element is out of use, that is, experiencing an outage. Availability is usually expressed as a percentage of time the element is not out of service; the availability measure is calculated by subtracting the duration of the outage from the base time and dividing the results by the base. High availability is the term usually associated with the ability to run for extended periods of time with minimal or no unplanned outage. In measurement terms, high availability is frequently considered to be 99.99% or greater. When looking at measurements of system availability, it is important to consider the context of that measurement. For example, a "5 nines" operating system/hardware combination does not mean that your systems will now only have five minutes of downtime in a year. Instead, this means that the operating system and associated hardware can achieve this. An entire system must take into account the network, software failures, human error, and a myriad of other factors. Thus, speaking of availability without a context does not necessarily have enough meaning.

Continuous availability is the ability to provide high levels of availability all day every day (24/7/365) with minimal planned or unplanned disruptions due to maintenance, upgrades, or failures. Figure 5-1 shows that no unplanned outages, yielding high availability, and no planned outages, yielding continuous operations, can combine to approach continuous availability.

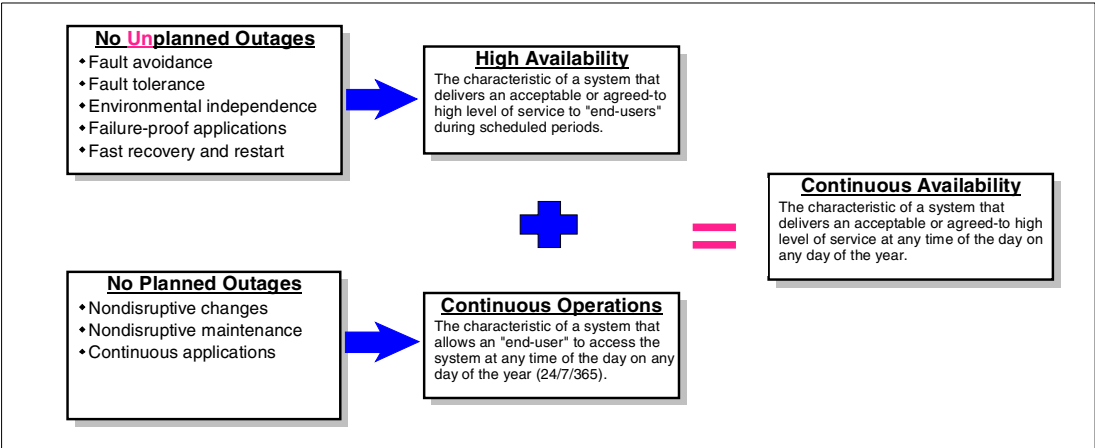


Figure 5-1 Continuous availability

Availability is best understood from the perspective of those who visit and use your Web site. From that perspective, three elements of an outage reduce availability: frequency, duration, and scope (the number of users affected). As you identify and implement availability improvements, you must understand what is contributing to the frequency, duration, and scope of your outages.

How long is an outage really? As a point of reference, here's how availability measures translate into actual time lost as shown in Table 5-1:

Table 5-1 Availability versus actual time lost

Measure	Time lost per year
99.9999%	32 seconds
99.999%	5 minutes
99.99%	53 minutes
99.9%	8.8 hours
99%	87 hours (3.6 days)
90%	876 hours (36 days)

To provide continuous availability, every element of your infrastructure must support continuous availability. The level of service delivered to your customers can be no higher than the availability of the weakest link. Improving one element to deliver near 100% availability while ignoring other elements does not provide as much benefit to your customer as a more balanced approach. Figure 5-2 shows an e-business infrastructure comprised of many elements with various availability measures.

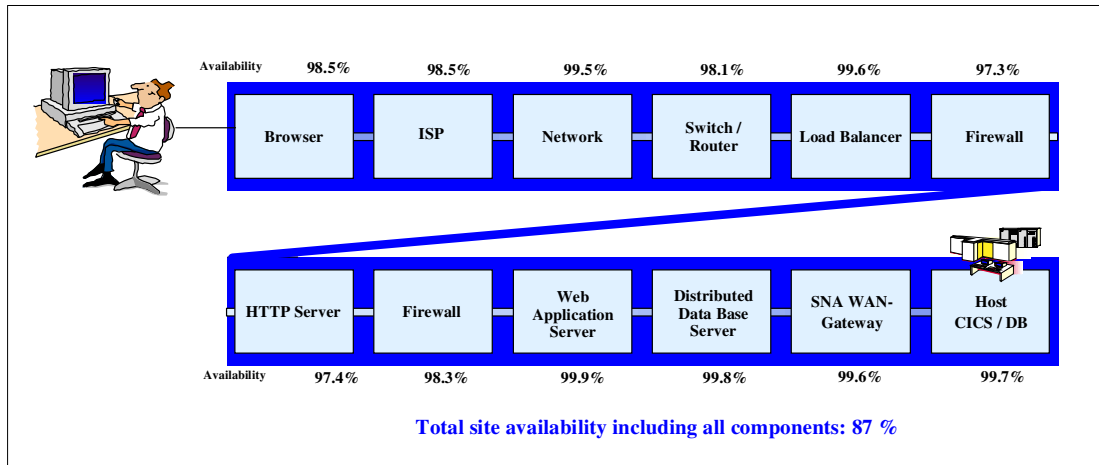


Figure 5-2 e-business infrastructure with availability measures

The total availability of an infrastructure is calculated by multiplying the availability of each component, for example, $98.5\% \times 98.5\% \times 99.5\%$ etc. It's great that many elements have pretty high availability, but the fact that some have considerably less and that the overall availability of the infrastructure is just 87%, suggests that a more balanced design would provide better total availability.

Unavailability is expensive

Outages, poor performance, and scheduled interruptions affect business opportunities, costs, and customer satisfaction. Outages bring tremendous costs to the business in the form of productivity losses in IT and the business units, lost revenue, and other penalties.

The cost of an outage is the sum of:

Productivity of affected users = hourly cost of affected users X hours of disruption

+Lost IT productivity = hourly cost of affected staff X hours of lost productivity

+Impact to customer service

+Lost revenue = lost revenue per hour X hours of outage

+Other business losses incurred

Overtime payments = hourly wages X overtime hours

+ Wasted goods

+ Financial penalties or fines

Typically, lost revenue represents the greatest financial exposure, but can be the most difficult to quantify.

Continuous availability is expensive

Availability is not free. It takes hard work and serious management attention to integrate many diverse components, people, and processes into a stable, highly available system. High availability starts with reliable products. Today's products are reliable and are capable of delivering high levels of availability, but reliable products alone will not assure high quality service. Very high levels of availability rely on an infrastructure and application design that includes availability techniques and careful system integration. A lack of, or failure to follow, careful management and effective systems management procedures, is actually the most

common cause of outages. Change management, in particular, needs more emphasis. Effective management systems that employ defined and repeatable processes contribute significantly to higher levels of availability while, in the long run, actually decreasing the cost of IT services through more effective and efficient use of IT resources.

Making trade-offs

While highly available systems are desirable, an optimum balance between the costs of availability solutions and the costs of unavailability is usually required. Factoring in the intangible consequences of an outage adds to the requirement to invest in very high levels of availability. Figure 5-3 suggests how to analyze the trade-offs between the costs and consequences of an outage.

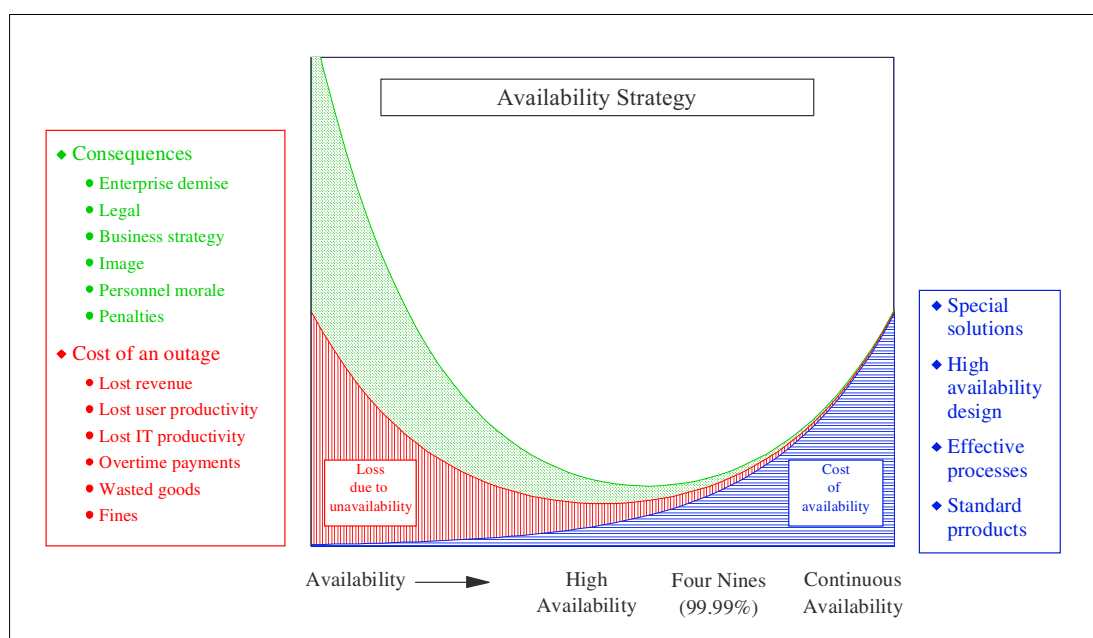


Figure 5-3 Trade-offs between costs and consequences of an outage

Figure 5-4 has a view of how this balance applies to different HVWS workloads and when the investment in availability could reach the point of diminishing returns. Each workload shown has a different cost to its business for each outage. Accordingly, each workload invests in availability in proportion to its outage costs. You can see that high availability becomes increasingly expensive as you approach 100%. The online broker, for example, will probably invest more than a marketing information site, but even its investment is eventually limited.

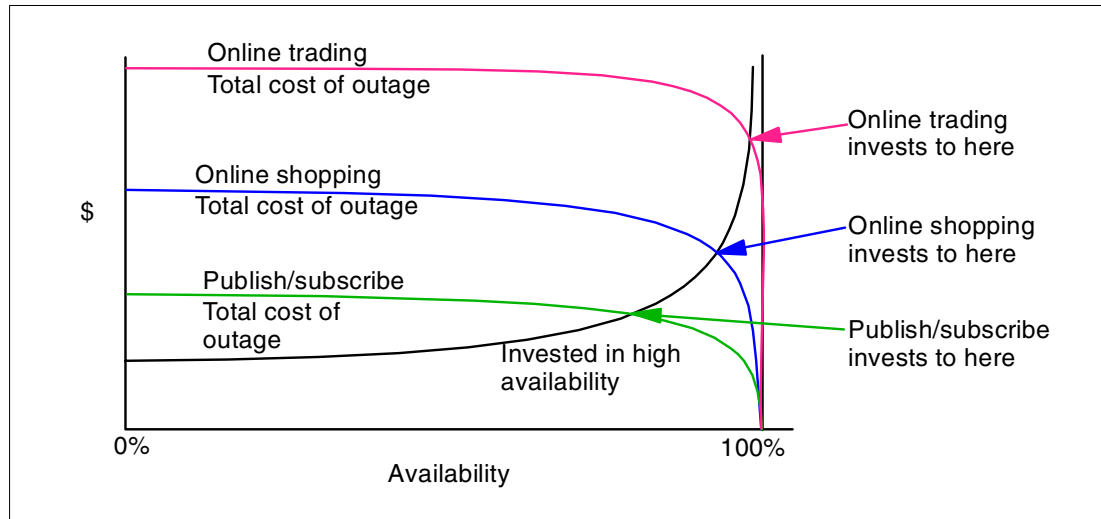


Figure 5-4 Investments in availability vary by workload pattern

On the way to continuous availability

This section introduces, at a high level, the factors that get in the way of achieving availability objectives and the common techniques used to improve availability. It concludes with a more detailed discussion of some of the improvement techniques. Of course, skilled employees are essential to achieving continuous availability. Your employees must be trained on the products, tools, and processes they use and/or support.

Common inhibitors

This list identifies the characteristics of your IT operation that may be contributing to outages and unsatisfactory availability. Your IT operation would be the exception if it didn't suffer at least one of these characteristics:

- ▶ IT focus is on component availability and tools rather than the users' perception of availability and processes to manage end-to-end availability
- ▶ At least one of the following could be considered insufficient:
 - Management focus
 - Long range planning
 - Change management
 - Problem analysis/management
 - Trained personnel
 - Process definition and repeatability
- ▶ Application design practices lag business availability requirements
- ▶ Failure to exploit availability features of current products

The Web environment introduces additional inhibitors that require at least understanding, if not proactive management to optimize for availability: ISP availability, client affinity, content management, DNS servers, and systems management. Most Web sites also face the challenge of an unpredictable workload. Sudden bursts of requests can exceed a site's

capacity and, at the least, increase the response time, sometimes dramatically, or, at the worst, cause the site to crash. Our HVWS white paper, Planning for Growth, describes workload burstiness and strategies to plan for it.

Management focus is required to significantly improve availability. There are many alternative solutions; the challenge is to understand the problems and customize the approaches to develop a solution.

Common techniques

This section introduces management and technical techniques you can use to reduce the number, duration, and scope of outages. Improvements can result from implementing any one, or any combination. High availability requires a balanced approach to implementing all.

Design your applications to enhance availability Application design must be mindful of objectives for continuous availability.

Assure your IT processes promote availability Effective systems management processes or management systems are necessary to proactively prevent problems, minimize risk associated with changes, and restore service quickly and efficiently.

Design your infrastructure for availability The primary characteristics of your infrastructure that can contribute to high availability are reliable components, redundancy of components and data, products with built-in availability functions and features, and automation.

The rest of this section discusses these techniques in more detail. Again, because of the complexities of the e-business environment, understanding and attending to these basics are more important than ever. Making the right choices in the design and management of your applications, your IT processes, and your e-business infrastructure will pay significant dividends in improved availability.

Design your applications to enhance availability

For an e-business Web site, the focus of application design must include the requirements of the customers for performance, availability, and reliability. Availability requirements must share the forefront of application design. Existing applications and processes may need to be redesigned with these increased requirements in mind. The consequences of ignoring or minimizing the requirement can no longer be tolerated.

IBM suggests your process for designing applications be enhanced to meet the availability requirement:

- ▶ Educate your designers and developers about customer requirements and the cost of an outage.
- ▶ Include availability objectives in application design; involve availability experts early.
- ▶ Exploit availability features of current products; maintain installed products at current levels.
- ▶ Select components that have characteristics suitable for the negotiated availability requirement.
- ▶ Consider data access and data maintenance.

- ▶ Design to keep the scope of failure small:
 - Minimize impact to other components
 - Isolate and contain important functions such as the session engine and databases
 - Provide selective redundancy for those functions that have broad impact to the rest of the system (for example, directories and databases)
- ▶ Design application recovery and initialization processes to be fast and easy on everybody (customers, operators, administrators).
- ▶ Seek alternatives to applications that require planned outages.
- ▶ Design batch processes that do not affect online applications.
- ▶ Employ standard processes and procedures to help improve communications and reduce the likelihood of errors. Naming conventions, for example, help reduce errors and can also provide a built-in check during changes and other activities.
- ▶ Acquire monitoring and problem determination tools that are robust and easy to use. Implement application-level monitoring as appropriate.

Robust testing is necessary to ensure conformance to availability design specifications and to validate that a component is ready to be introduced into the production environment. A complete end-to-end system is a unique integration of many diverse hardware components, software systems, subsystems, applications, network, servers, and access devices assembled from various vendors and suppliers. Documentation, procedures, and numerous support organizations are also involved in delivering quality service. These must be tested together in a system test or they will be tested for the first time in production with possibly disastrous consequences.

Assure your IT processes promote availability

Your availability measures are true indicators of the scope and effectiveness of your IT processes. In the large e-business infrastructures we work with, the presence of thorough, well-understood, and well-managed processes remain a key factor in site availability. Sadly, it often takes an unplanned outage to put the spotlight on insufficient processes.

IBM suggests you review your IT processes to be sure that the availability requirement is reflected:

- ▶ Ensure IT vision includes availability; many organizations will contribute to the success of your vision; assure each understands the significance of an end-to-end approach and of an inclusive, efficient process of communicating among organizations
- ▶ Create, maintain and manage the availability strategy and plan
- ▶ Establish target objectives
- ▶ Monitor service level achievements
- ▶ Monitor end-user availability, identify availability issues, and follow up
- ▶ Sensitize the organization to the value of availability and lead your IT operation to a proactive environment
- ▶ Assess current state and identify gaps
- ▶ Monitor and guide these related and dependent processes for availability considerations:
 - Problem management
 - Change management
 - Recovery management
 - Capacity management, including planning for traffic bursts

- Configuration management
 - Design, development, and deployment processes (phase reviews, availability guidelines)
 - Testing
- Ensure architecture and design includes availability objectives and features

Web-based applications often span a number of different resources, from the front-end to back-end legacy systems. It is critical to have a systems management solution that can consistently manage and secure the total environment. There are many management products that function well to address individual resources. However, if Web site availability is critically important, a complete end-to-end solution can best meet your requirements. A solution from Tivoli can address each of the following critical areas.

Manage availability from the end-user's perspective. Any systems management solution for improving Web site availability should begin with the end goal in mind. The experience of the actual end-user accessing your Web site should be monitored regularly. If the user experience begins to fall below what is considered acceptable, administrators must be notified and corrective action taken immediately.

Availability in this context refers to not only the traditional definition of ensuring that a user does not encounter Page Not Found (404) errors, but also two broader factors:

1. Users must be able to get online and be productive quickly, for example, by automating the process of signing up users and enabling their access to appropriate resources, after which the e-business is available to them
2. When users are online, the content delivered to them meets predetermined performance guidelines, for example, considering the time it takes for a user to access pages on the Web site. Users perceive slow sites to be unavailable.

Consider the end-to-end business process view. Resources must be managed in the context of a business process. Ensuring Web site availability is based on several different resources acting together as an end-to-end system. An advanced systems management product provides the topology view of the resources that make the business processes associated with a given Web site. This allows administrators to understand the impact of any one resource on the Web site and ultimately allow administrators to prioritize their actions based on real business needs.

Ensure availability through consistent policy compliance. A leading inhibitor to availability is unexpected downtime from security breaches, which are typically caused by inconsistent application of a security and privacy policy across the entire application and platform stack. Because application developers, IT security administrators, and IT operations groups usually manually administer security and privacy policies across only the resources that they directly manage, simple policy actions such as adding or deleting customers (or employees or partners) often result in inconsistent levels of access being set across the infrastructure. The ability to set the security and privacy policy once, and have that policy consistently and immediately applied across the entire e-business stack, is a critical success factor for availability.

Monitor availability for all critical infrastructure resources. If it is determined that the user experience has started to deteriorate, systems administrators must have tools to understand the health of the secure, critical infrastructure on which your Web site is built. This infrastructure includes all the resources that have a part in the site's availability. In a WebSphere environment, this should include front-end resources such as Web servers, the WebSphere Application Server, messaging software such as MQ Series, as well as the back-end legacy applications that might reside on zSeries and other platforms. The entire

stack should be considered across these resources, including the hardware, operating systems, network security devices, middleware, and the applications themselves.

Ensure optimal configuration of your system resources. Tools are available to configure and deploy software and upgrades thereto across your infrastructure. This assures that designated components are equipped with the same level of software and avoids outages that result from incompatibilities.

Automate workload scheduling across your infrastructure. Tools are available to automate tasks and/or series of tasks. This assures that tasks are completed when and where required and avoids outages that result from missed steps.

Capture and make use of historical management data. A thorough systems management solution captures a considerable amount of data from a variety of perspectives as outlined above. Not only is this data important for understanding and ensuring the real-time availability of your Web site, but it also serves as valuable input for historical reporting applications. Such applications should include functions such as service level management, capacity planning, and security audit compliance. Ideally all the management data captured should be sent to a common data warehouse so that customers may compare and correlate as required from the variety of sources available. Such an approach will allow you to discover trends and plan for sufficient capacity to maximize high Web site availability as your needs change over time.

Design your infrastructure for availability

The typical e-business Web site has an infrastructure comprised of several tiers from the firewall(s) to data/transaction server(s). There are usually two or three, less commonly four core tiers that handle a particular set of functions, such as serving content (Web servers), providing integration business logic (Web application servers), and processing database transactions (transaction and database servers). A multi-tiered infrastructure inherently contributes to availability by segregating functions and simplifying communications among components of the infrastructure.

Reliable components are key to availability. Hardware components can fail and software quality varies from release to release, making other techniques for high availability equally important. Availability functions and features are built in to most products. Some features are automatic and are effective by default, but others require exploitation and tuning. Effective installation, implementation, tuning, education, and documentation processes are imperative to be sure full advantage is realized from these features.

Redundancy is a primary and well-understood means of achieving high availability -- redundant components, redundant systems, redundant data, and even redundant people. Redundancy works by eliminating single points of failure. There are many approaches and levels of redundancy, each with different cost and effectiveness trade-offs. Figure 5-5 provides an example of a multi-tier system with redundancy built-in to each layer to provide high-availability.

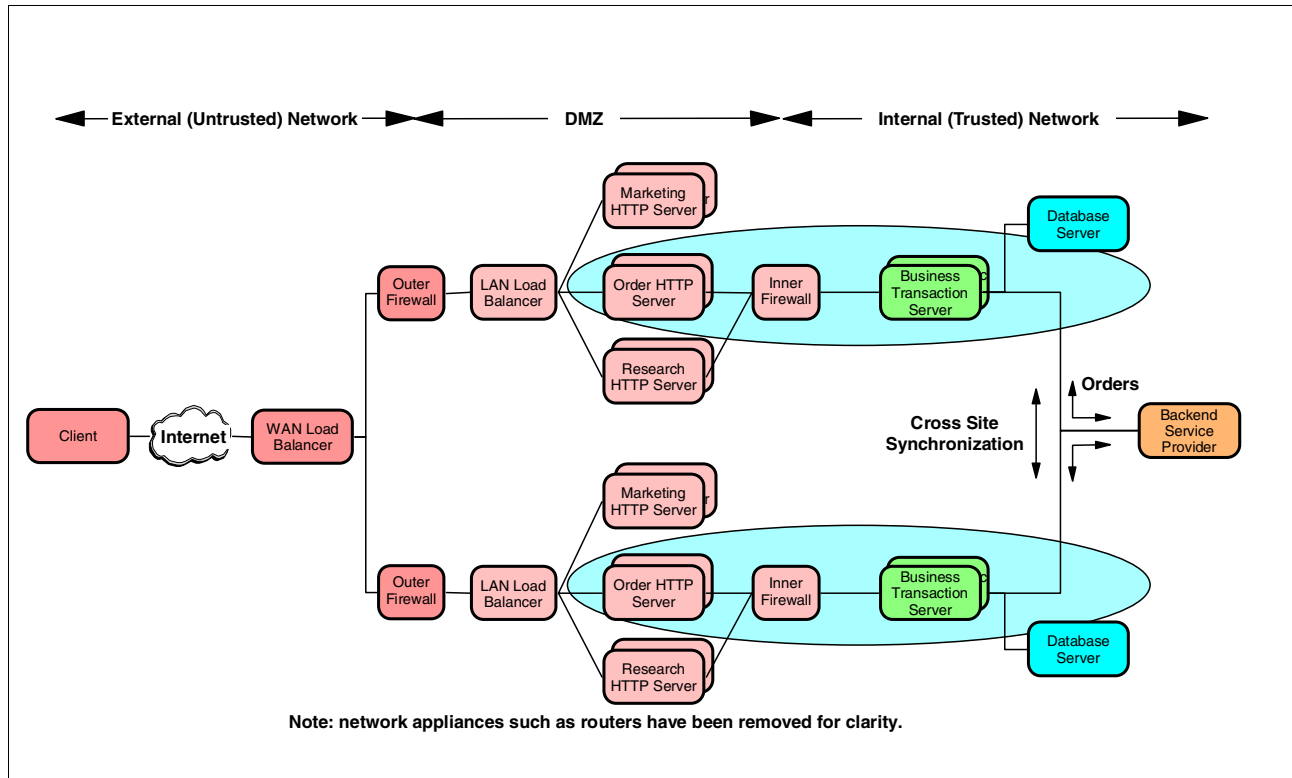


Figure 5-5 Multi-tier infrastructure designed for high availability

Automation is critical to continuous availability. Time lost during a failover to back-up diminishes availability. Automation is necessary to reduce or eliminate human intervention during recovery. Automation also introduces consistent execution and reduces the probability of human error.

This section reviews techniques that can improve the availability of your overall site infrastructure as well as that of specific site components. It presumes you seek reliable components with built-in availability features and employ them as soon as and as much as possible. It concludes with techniques for those complex infrastructures that use more than one physical site.

Create hardware and software clusters / implement load balancing

A cluster (either hardware or software) provides the infrastructure to allow higher availability and scalability of any given component. The basic tenant is to provide a common addressing scheme to various underlying components. For example, a load balancer/IP sprayer provides a single IP address/name for a group of servers. The load balancer/IP sprayer routes requests to the appropriate server within the cluster. End users only need to address their requests to the published IP resource; the load balancer determines which server should handle the request. The load balancer should also be able to determine when a server in its cluster is functioning (or not), and route requests only to functioning servers. This provides greater availability for the given resource. A second benefit is the ability to add (and subtract) servers from the cluster, which allows the cluster to scale to meet end user demands for the service. Clustering can be implemented in each tier to the extent needed to optimize for availability. When clusters are used, load balancing techniques are used to dynamically manage site traffic. Specific load balancing components are often used to distribute network traffic across the servers based on criteria such as load equalization, server utilization, and/or application affinity.

Typically, redundancy is implemented by installing one or more components that are identical to the primary component. Each component has different availability probabilities; how you combine them is key to the overall environment availability.

When redundancy is applied to components with a high probability of failure, for example, the Web server, the overall availability for the aggregate of redundant components increases. Depending on the component's function, there are specific considerations regarding redundancy and load balancing. Table 5-2 summarizes considerations for specific components.

Table 5-2 Components for consideration

Component	Consideration for redundancy and load balancing
Exterior connectivity	Multiple Internet service providers (ISPs) assure that an ISP failure does not make your site unavailable
Domain name server (DNS)	Use one primary and any number of backup DNSs. A DNS supports multiple sites by passing a different IP address every time it is queried with a name (DNS round robin selection).
Load balancer	Implement a backup load balancer. The backup would be a similarly configured load balancer that has a heartbeat with the primary; if the primary load balancer fails, the backup initiates a take over function. This feature is available in most load balancers and has been implemented by the WebSphere Edge Server.
External cache	Implement redundant caches in parallel with a load balancer to distribute the load. In a reverse proxy scenario, this increases availability of the Web servers.
Firewall	Implement redundant firewalls, load-balanced to give the appearance of being available. Because the connections through a firewall tend to be longer (socket connections versus HTTP), the load-balancing decision needs to be made when the session is initiated and kept during the whole session.
Web server and application server	Implement redundant servers, load-balanced to give greater availability. Each server should be able to handle identically any request. If a shopping cart is being used, some sort of persistent store is needed. If the server doesn't have the user's state cached, it must get it from the persistent store.
Directory and security server	Implement redundant servers, load-balanced to give the appearance of being available for read only operations. For write operations, use a primary server and replicate changes to the backup machines.
Databases	Implement physical or logical failover and recovery schemes for application, administration, and session databases.
Physical connections between servers	Implement redundancy of the physical connections.

WebSphere considerations The WebSphere Application Server is architected to provide clustering. With prudent planning and deployment, the WebSphere runtime (and your applications) can be made highly available. When planning for high availability with WebSphere, consider redundancy for the components listed in Table 5-2. Give special consideration to:

- ▶ WebSphere Application Server (both Web container and EJB container)
- ▶ WebSphere Administration Server (security, naming (bootstrap and name server), location service daemon, workload management, serious event log, transaction log)
- ▶ The database containing the WebSphere Administration Server; configure this database server with a hardware-based solution such as high availability cluster multi-processing (HACMP).

Figure 5-6 shows a minimal topology for a highly available WebSphere configuration. For more information, refer to the paper Failover and Recovery in WebSphere Application Server Advanced Edition 4.0.

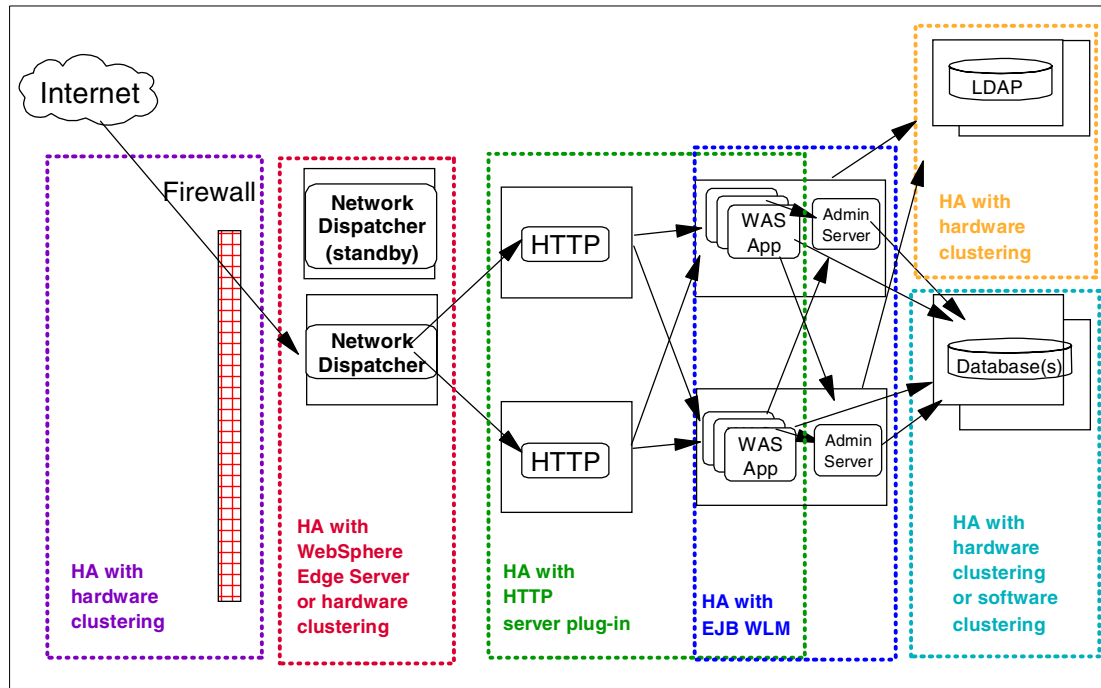


Figure 5-6 Minimum topology for a highly available WebSphere configuration

Consider WebSphere Edge Server for Multiplatforms

WebSphere Edge Server for Multiplatforms is Web infrastructure software that addresses the scalability, reliability, and performance needs of e-business applications in local and geographically distributed environments. Its functions incorporate robust, leading-edge caching and load balancing that together compensate for the inherent weakness of the Internet to support critical business applications and expectations.

Network Dispatcher (ND) is the load balancing component of the WebSphere Edge Server. It is a software based solution that provides dynamic load balancing across multiple servers, groups of servers, and geographic locations in a highly scalable, highly available fashion. As a software solution, it is highly adaptable to an enterprises' particular needs. ND dynamically balances not only HTTP traffic, but SSL and FTP as well (which can be major components of a Web site's usage pattern). Through its use of advisors (Java code that executes from a client's view of the Web site), ND can detect the status of servers beyond the first tier of HTTP servers. It has the ability to follow a code path through the HTTP server, to the application servers, and through to the back-end database servers. If it detects that the resources on which the HTTP servers depend are having problems, it routes traffic around those HTTP servers, thus providing a better end use experience of the Web site. Built into ND is a high availability feature that involves the use of a secondary machine that monitors the main, or primary, machine and stands by to take over the task of load balancing should the primary machine fail. This rapid takeover maintains the open connections that have been created on the primary ND. This technology (ND) has been used to power some of the busiest sites in Internet history. The Olympic Web sites, Wimbledon and U.S. Open Web sites, and many others, which were processing millions of hits an hour, have all been powered with ND providing the dynamic load balancing.

Web Traffic Express is the web caching/proxy server portion of WebSphere Edge Server. A caching proxy server is useful for enterprises wishing to enhance the Web experience for people visiting their Web sites, because it pushes the content closer to the end users, thus giving them faster response. With the latest release of the Edge Server (Version 2.0), the ability to store static content (as traditional caching proxies do), has been augmented by the ability to actually execute application software on the caching proxy. This functionality not only allows for increased availability (now there are more servers that can handle the requests), but also adds scalability to the Web site (because the back-end servers are handling fewer requests).

Consider availability while implementing connectors

The IBM WebSphere MQ family provides an open, scalable, industrial-strength messaging and information infrastructure, enabling enterprises and beyond to integrate business processes. MQSeries provides asynchronous messaging, thereby enabling connections among multiple systems. The central issue in availability is that, when possible, messages can take alternate routes through the MQSeries infrastructure and not be delayed for longer than necessary where alternate routing is not possible. The availability of an MQ solution can be increased by hardware clustering, software clustering, queue sharing, and fault tolerant operating systems. These techniques provide failover, simultaneous access, or hardware redundancy to maximize the availability of even a single queue. Software clustering and queue sharing exploit multiple queue managers to achieve higher availability.

Connection pooling is an application-server concept used to minimize the amount of resources used for connections to database and transaction servers. If the resources required to support a particular connection, here called a connection agent or simply agent, are relatively large, it makes sense to maintain an agent for only the active connections. Further, if it is relatively expensive in terms of processing to construct and destroy an agent, it also makes sense to keep some around for immediate assignment to connections that make the transition from inactive to active. In this technique, when a connection transitions from active to inactive, its agent is “cleaned up” and placed in the pool of such agents ready to be assigned to connections that become active. This ready pool of agents is the connection pool. Connection pooling can be used by any type of server to reduce the cost of supporting connections going from inactive to active to inactive. Your infrastructure should include a mechanism to recover the connection pool from a disruption of service to the database server. WebSphere Application Server includes such a mechanism.

Protect your data from accidental loss, viruses, and disasters

It is critical to protect and manage your mission-critical business data, by implementing a centralized enterprise backup, archive and disaster recovery solution over a storage area network (SAN) and/or traditional network environment. A solution like Tivoli Storage Manager can enable you to restore and recover files, directories, file systems, databases, or entire computers after the accidental deletion of a document, virus infection, hardware failure or theft, or destruction of your IT center due to some sort of disaster.

While many companies place special demands on their storage management software, fundamental expectations apply across all environments. These expectations are that storage management software provides:

- ▶ High data availability
- ▶ Backups that have limited impact on systems in use
- ▶ Rapid, easily available, and completely reliable data recovery
- ▶ Clear and precise disaster recovery capability to aid in a disaster

The relative importance of these items varies among companies, but all need to appear in a company's assessment of its own efficiency of operation.

SANs change how data is accessed and increases availability

A SAN is a dedicated, centrally managed, secure information infrastructure, which enables an any-to-any interconnection of servers and storage systems. A SAN:

- ▶ Facilitates universal access and sharing of resources.
- ▶ Supports unpredictable, explosive information technology (IT) growth.
- ▶ Provides affordable 24 x 365 availability.
- ▶ Simplifies and centralizes resource management.
- ▶ Improves information protection and disaster tolerance.
- ▶ Enhances security and data integrity of new computing architectures.

SANs are based on a systematic approach to data storage management pioneered by IBM almost 30 years ago. Now SANs are rapidly being integrated into distributed network environments using fiber channel technology. SANs improve availability because of their abilities to dynamically reconfigure storage environment to handle increasing capacity requirements or to recover from hardware failure, and to share a single copy of data among multiple host servers. Their enhanced copy functions facilitate the "cloning" of servers locally or in geographically dispersed sites.

Multiple physical sites to protect against the effect of disasters

For customers who require very high availability and reliability from their e-business Web sites, locating the Web servers in multiple geographically-dispersed sites protects against potentially catastrophic events such as:

- ▶ Complete power failure of computer complex
- ▶ Loss of network connectivity due to failure at network service provider premise
- ▶ Overloading of network capacity at a single site due to demand peaks or denial of service attacks
- ▶ Physical disasters

Properly implemented, a multisite front end has significant benefits:

- ▶ High availability/reliability
- ▶ Load balancing of network traffic between sites
- ▶ Potential for "smart" caching functions

At the same time, it adds another level of complexity to the overall Web solution in trying to balance the network traffic among the sites and provide seamless redundancy. A plan for a multisite implementation must include a strategy for failover and load balancing. The loads at each site must be managed so that sufficient capacity is available if an event requires a site to assume the load of a failed site.

Summary

Today's high volume e-business Web sites require the highest possible availability. Increasingly, e-business sites are demanding continuous availability. No link in the availability chain can be broken and the central engine of commerce, the server, cannot be down. Many corporate strategies demand 7/24/365 availability of service and the IT organizations must support this strategy.

As with nearly all the traditional IT challenges -- design, performance, capacity planning -- the availability challenge hasn't necessarily gotten any easier in the e-business world. We learn again that the fundamentals are as important as ever. And we learn of new and improved products and techniques that can help address the constant challenges. These best practices can contribute to your availability objectives:

- ▶ Design your application to enhance availability
- ▶ Assure your IT processes promote availability
- ▶ Design your infrastructure for availability

Availability is a discipline, it is a technology, it is a corporate culture. It involves systems management, application design, even organizational structure and above all, a commitment from the IT organization and the business to make it a focus. There is no free lunch. In this chapter, our objective has been to help you understand how you can achieve world class results and to arm you with knowledge so you are the informed decision maker in this critical area of IT management.

References

Failover and Recovery in WebSphere Application Server Advanced Edition 4.0. See:

<http://www7.software.ibm.com/vadd-bin/ftpd1?1/vadc/wsdd/pdf/modjeski.pdf>

So You Want to Estimate the Value of Availability, IBM publication GG22-9318

See the following IBM Redbooks at:

<http://www.ibm.com/redbooks>

WebSphere Edge Server: Working with Web Traffic Express and Network Dispatcher,
SG24-6172-00

WebSphere on RS/6000 SP and Clusters Centralized Management and Monitoring,
SG24-6037-00

Roadmap to Availability on the iSeries 400, REDP0501

Carnegie Mellon Software Engineering Institute at:

<http://www.sei.cmu.edu>



Part 2

Customer engagements

Part 2 describes customer engagements that characterize the type of work the High-Volume Web Sites team performs for its customers.



Managing Web site performance

As more of your company's business moves to the Internet, your IT organization is becoming a major focal point for such important business measures as revenue and customer satisfaction. You're enjoying unprecedented visibility, and it may not all be positive. If it hasn't already, the performance of your Web site will become critically important.

This chapter deals with managing performance. More than ever before, this task requires a perspective that considers components of the infrastructure from end-to-end: from the front-end browsers to the back-end database servers and legacy systems. The end-to-end perspective must be shared not only by you and your operations staff, but also by application developers and Web site designers. Required as well are thoughtful objectives for performance coupled with thorough measurements of performance.

This chapter proposes a methodology that you can follow to manage your Web site's performance from end to end. Ideally, you have characterized your workload, selected and applied appropriate scaling techniques, assured that performance is considered in Web page design, and implemented capacity planning technologies. If you have not, you may want to review the white papers related to those phases of the life cycle at the same time as you consider this methodology (see References). Regardless, the methodology presented here can help you define your challenges and implement processes and technologies to meet them.

Our best practices methodology for managing the performance of a high-volume Web site consists of familiar tasks:

- ▶ Establish objectives
- ▶ Monitor and measure the site
- ▶ Analyze and tune components
- ▶ Predict and plan for the future

Some benefits you can expect after implementing the end-to-end methodology include:

- ▶ Proper reporting of quality of service metrics
- ▶ Interactive and historical data on end-to-end performance
- ▶ Rapid identification of the problem source

- ▶ Improved support of business goals
- ▶ Understanding and control of transaction costs
- ▶ World class customer support and satisfaction

The goal of implementing the end-to-end methodology is to align the system performance with the underlying business goals. The methodology, coupled with implementation of the capacity-on-demand options available from IBM's powerful server family, make the goal achievable, and set the stage for self-managing IT infrastructures.

Managing the performance of a high-volume Web site requires a new look at familiar tasks such as setting objectives, measuring performance, and tuning for optimal performance. First, HVWS workloads are different from traditional workloads. HVWS workloads are assumed to be high-volume and growing, serving dynamic data, and processing transactions. Additional characteristics that can affect performance include transaction complexity, data volatility, security, and others. IBM has determined that HVWS workload patterns fit into one of five classifications: publish/subscribe, online shopping, customer self-service, trading, or business-to-business. Correctly identifying your workload pattern will position you well for making the best use of the practices recommended in this and related chapters. For more information about how IBM distinguishes among HVWS workloads, see the Design for Scalability white paper.

Secondly, those performing the tasks must extend their perspective to include the e-business infrastructure from end to end. This is most effective when all participants understand the application's business requirements, how their component contributes to the application, and how a transaction flows from one end of the infrastructure to the other. Only then can they work together to optimize application performance and meet key business needs. It's often best when one person is assigned ownership of each application considered critical to the e-business; the application owner assures the customer's perspective of application performance -- response time -- remains the primary focus of all participants.

This chapter proposes a methodology that you can follow to manage your Web site's performance from end to end. Ideally, you have characterized your workload, selected and applied appropriate scaling techniques, assured that performance is considered in Web page design, and implemented capacity planning technologies. If you have not, you may want to review the white papers related to those phases of the life cycle at the same time as you consider this methodology. Regardless, the methodology presented here can help you define your challenges and implement processes and technologies to meet them.

Figure 6-1 shows our methodology for managing the performance of a high-volume Web site in the context of a multi-tier infrastructure.

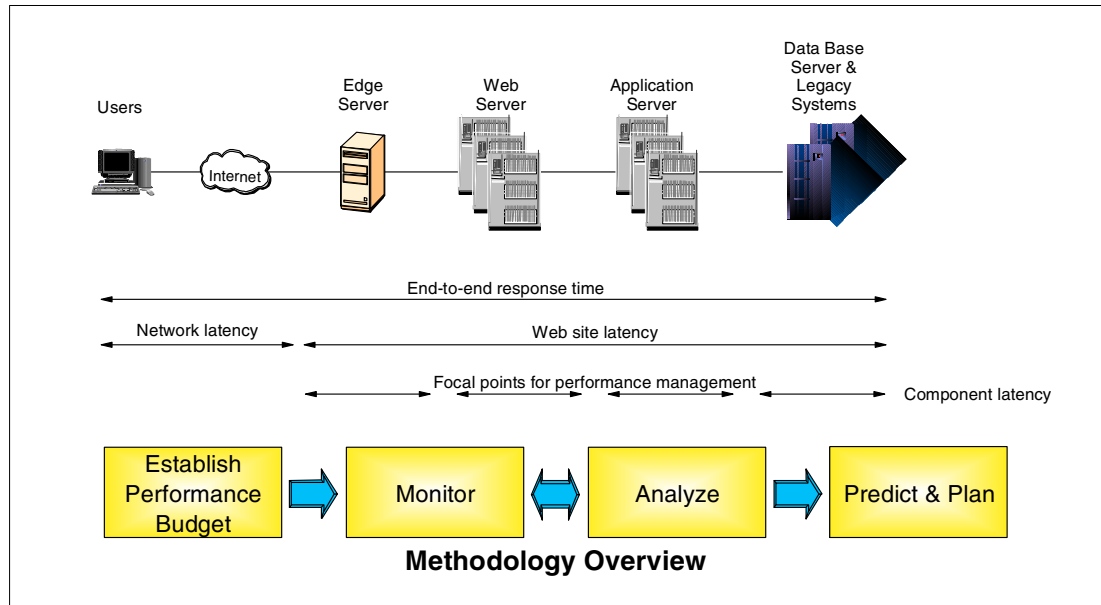


Figure 6-1 Methodology for managing performance of a HVWS

Our methodology consists of familiar tasks with a new twist, driven by the requirement for an end-to-end perspective, and including tools that are available now to help you get started. The last part of this chapter has some sample scenarios about managing performance and a summary of tools available from IBM, including Tivoli, IBM's provider of e-business infrastructure management software.

Step 1. Establish performance objectives

The first task is to establish performance objectives for the business, the application, and operations. Performance objectives for the business include numbers of log-ons and page hits, and browse-to-buy ratios. Objectives for the application include availability, transaction response time, and total cost per transaction. Operations objectives include resource utilization (network, servers, etc.) and the behavior of the components that comprise the e-business application.

You should use the results of an application benchmark test to establish the "norms". Ideally, you acquire the norms from controlled benchmark and stress testing. If this isn't possible, you should closely monitor and measure the deployment of the application and use the results to produce a performance profile ranging from the average to the peak hours and/or days.

Metrics should be established from outside the site (response times, availability, ease of navigation, security, etc.), and from each server tier (CPU, I/O, storage, network utilization, database load, intranet traffic rates, etc.). You need to establish thresholds so that operations can be notified when targets are near, at, or over their limits. The last section of this chapter has a list of tools available from IBM and Tivoli.

Managing against a set of norms is an ongoing process. Frequent updates to expectations and thresholds may be required. Marketing may schedule a promotion that will drive site traffic to new highs. It is important that this be planned for to avoid "false alarms" that can occur if you haven't updated your thresholds for the expected spikes in load.

The team that sets the objectives should include representatives of each area; if that is not possible, the combined objectives should be communicated clearly to all areas, along with the emphasis on what may be considered a new paradigm, that of the end-to-end perspective.

Step 2. Monitor and measure the site

In this step you examine and analyze the performance of the application. You view the application as a transaction flow from the browser through the Web servers and, if applicable, to the back-end database and transaction servers, and back to the browser. You are concerned with the entities that make up the system (operating system, firewalls, application servers, Web servers, etc.) only insofar as they support the application.

To understand end-to-end performance, you must understand and document the flow of each transaction type, for example, search, browse, buy, trade, etc. That done, you can use software that monitors the actual flow and alerts operations when any metric you specify exceeds the norms established in Step 1.

For example, the alert informs you that your target page response time has been exceeded. You know that something in the system has degraded, but where is the slowdown occurring? How do you find the culprit?

You could instrument your application to record information at various points in the transaction flow. An open standard, ARM (Application Resource Management) defines an API and library for these records. In addition to Tivoli, several vendors have tools to display and analyze this data. We have used exactly this type of instrumentation to manage various high-volume Web sites. It is important to note that instrumentation adds modifications and overhead to the application.

Instead of recording information on every transaction, you can take averages at several points. Information about averages is nearly as good as full instrumentation, but comes at a lower cost and uses existing and transparent tools. Tools such as the IBM Tivoli Performance Viewer can be used to extract these averages through the resource management interface.

Other available tools:

- ▶ Report on the quality of customer experiences
- ▶ Analyze the Web site to verify links and enforce content policy
- ▶ Aggregate Web data into an overall business view
- ▶ Correlate log and performance data
- ▶ Monitor availability
- ▶ Use online analytic processing (OLAP) techniques to provide decision support

WebSphere Application Server provides a set of comprehensive performance metrics. For servlets and beans, these include: number of requests, requests per second, execution time, and errors. Java™ metrics reported include: active memory, available memory, threads active, threads idle, etc. Database connections are also included: connection times, active database connections, users waiting for database access.

It's best to continuously monitor the site availability from outside to insure that transactions are executing successfully and within criteria. Examine the site navigation periodically to validate the links and content. Resource monitors will need to roll up their data into an aggregate application view. Web logs have to be analyzed and correlated with other resource data.

In a recent customer engagement, IBM's HVWS team investigated a problem of slow customer response time. The Web server was running Netscape Enterprise Server and the WebSphere Application Server for dynamic content generation using Java servlets. A middle tier used enterprise Java beans (EJBs) to process transactions and then a JDBC call to the database tier. Using the external monitor for response times, they found that during peak hours, response times for consumer transactions increased from fourteen seconds to twenty seconds. Using the IBM Tivoli Performance Viewer and some DB2 tools, they collected the internal elapsed times for the application components. Figure 6-2 shows the analysis of how each component contributed to total response time. Comparing the baseline time with the peak times, it's easy to see that the slowdown occurs in the servlet tier. Performance Viewer showed that the application server was running out of worker threads under peak load. Allocating additional worker threads eliminated the slowdown.

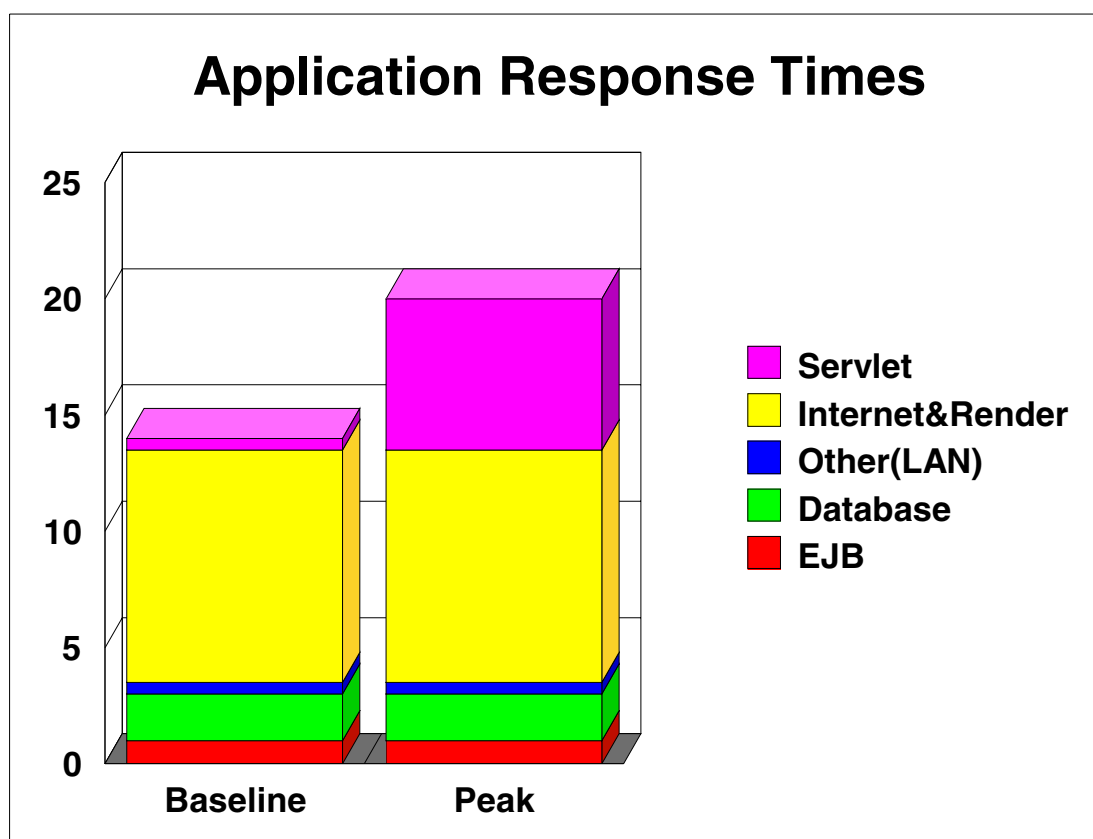


Figure 6-2 Application response times - baseline vs peak

Step 3. Analyze and tune components

So far, the methodology has provided objectives, measurements, and application insights. Thus it has allowed you to understand, monitor, and report on end-to-end performance. It has also allowed rapid problem determination. When performance issues come up, you can quickly investigate the application and isolate an individual component. Scenarios based on real events that show how components are analyzed and tuned are described at the end of this chapter.

In this step, you analyze and tune specific components. One common question is Does the application scale gracefully? In general, scalability refers to a component's ability to adapt readily to a greater or lesser intensity of use, volume, or demand while still meeting business

objectives. You want to assure that your application scales smoothly wherever deployed without experiencing thrashing, bottlenecks, or response time difficulties. You need to examine how your application uses resources: you're interested in CPU consumption per transaction and I/O and network overhead. See also Design for Scalability, our HVWS paper that recommends which scaling techniques should be applied to specific components.

Another important question: Is the application meeting economic criteria? Now that resource consumption is understood, you know the 'cost per transaction' and you can assess whether the application is using resources as projected by the performance objectives. You want to consider the best practices pertaining to scalability and page design and learn what's needed to optimize how the resources in each tier are used. The application owner uses this to work with development, operations, and design to control and/or improve the efficiency of the application. In this way costs are held on budget.

We used the methodology recently to benchmark a customer's application and found that throughput seemed to be stalled in the database server. Furthermore, the database was consuming more resources than was expected based on the historical archived data. The DBA ran the analysis tools and quickly determined that one of the application SQL statements was forcing a full table scan (very expensive, very bad). This hadn't had any measurable effect during the initial deployment of the application with a limited number of customers. However, as the number of customers grew, the size of the database increased significantly. The DBA was able to define an alternate index into the table, test the change, and resolve the problem within a short time. It was the methodology that pointed us quickly to the database tier and allowed us to determine the cause of the problem and solve it quickly.

The all-important question Can response time be improved? Using the component response times, the application owner works with operations to tune and allocate resources to insure good response times. For example, the Web servers may need more memory to allow a larger cache and reduce I/O times.

In one recent engagement, the customer help desk was flooded with complaints of slow or nonexistent performance. The senior management was concerned that the system seemed to be failing and IT seemed unable to tell them why. Using our methodology, we accessed the site with the WebSphere Studio Page Detailer to analyze page response times. Page Detailer showed us that response times were long due to excessive delays in obtaining TCP/IP socket connections. We investigated the intranet, firewalls, and site connectivity. It turned out that when the site went online, the firewalls had been set up to allow a fixed number of concurrent socket connections. As traffic increased (the site was succeeding), more and more customers contended for the same number of connections. This was easily corrected. In this case, as in many others, the solution seems obvious when you isolate the fault to an individual component. It is the methodology that allows us to do so.

Figure 6-3 shows tools and technologies available to monitor and analyze Web site components. You can see, for example, that you can monitor response time proactively using WebSphere Studio Page Detailer and IBM Tivoli Monitoring for Transaction Performance. Tivoli provides Monitoring for Web Infrastructure to manage and monitor Web server and application server components. The last section of this chapter has more detail about some available tools.

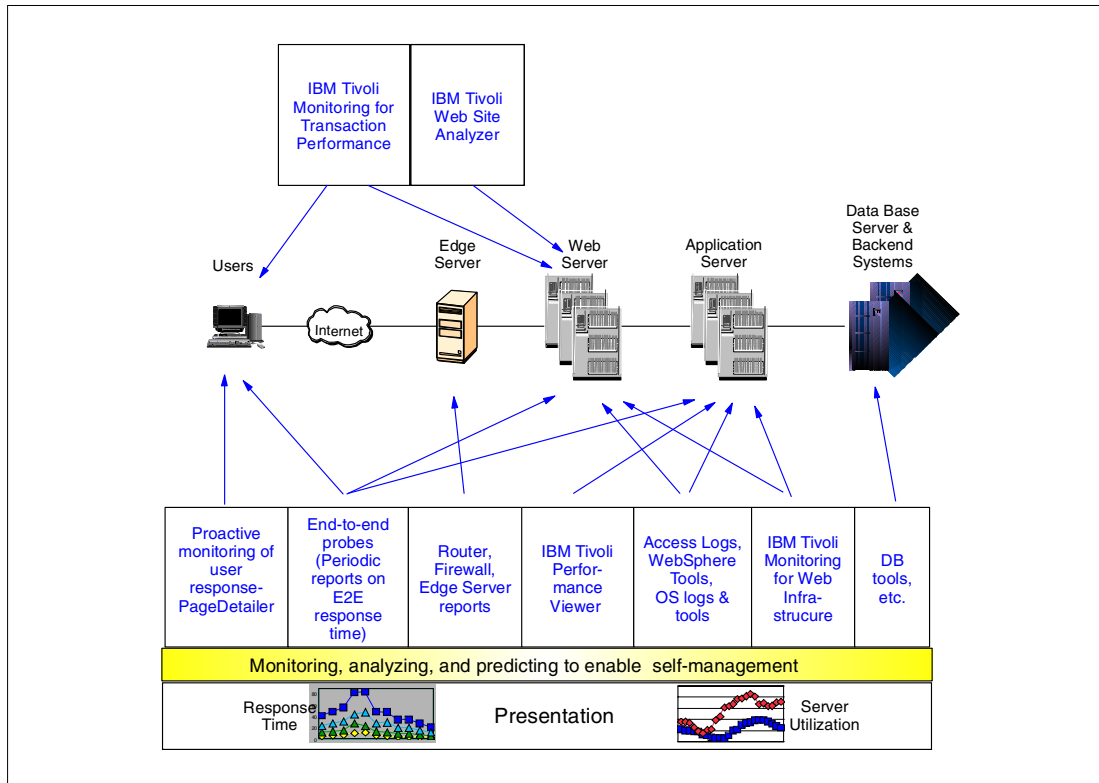


Figure 6-3 Tools available to monitor and analyze Web site components

Step 4. Predict and plan for the future

Sadly, none of us can predict the future. However, an increasing amount of valuable information and useful tools are available to help you plan proactively to keep your Web site serving customers as they expect to be served and to avoid the problems that plague busy sites.

Figure 6-4 shows one week of page hits for one of IBM's retail customers. All of the days have essentially the same pattern with predictable peaks and valleys. This site showed no 'weekend effect', which may not be true for its 'brick and mortar' store, nor for other retailers. This kind of information enables site personnel to prepare for peaks and use the valleys for other operations when needed.

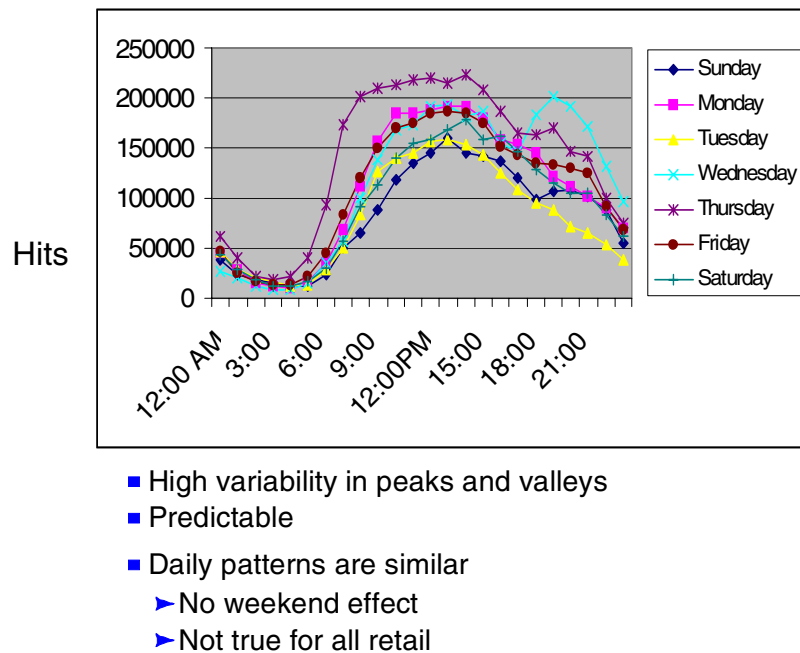


Figure 6-4 Retailer usage patterns over one week

While a typical week, as shown in Figure 6-4, can be counted on, a retailer also has to plan for seasonal rushes when peaks can easily exceed those of a typical week. Figure 6-5 shows a retail site over six months, including the annual holiday period when the number of hits tripled. During this kind of load, the site must be at its best, if possible free of other operations.

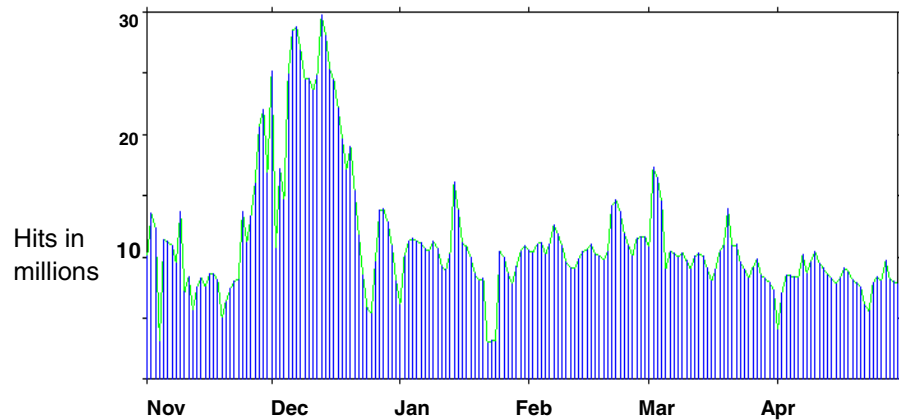


Figure 6-5 Retail customer seasonal peaks

Retailers aren't the only e-business facing seasonal demands. Look in Figure 6-6 at how the number of hits for a bank grew over the months approaching tax time. Clearly, the financial sites have their own version of weekly and seasonal peaks and valleys.

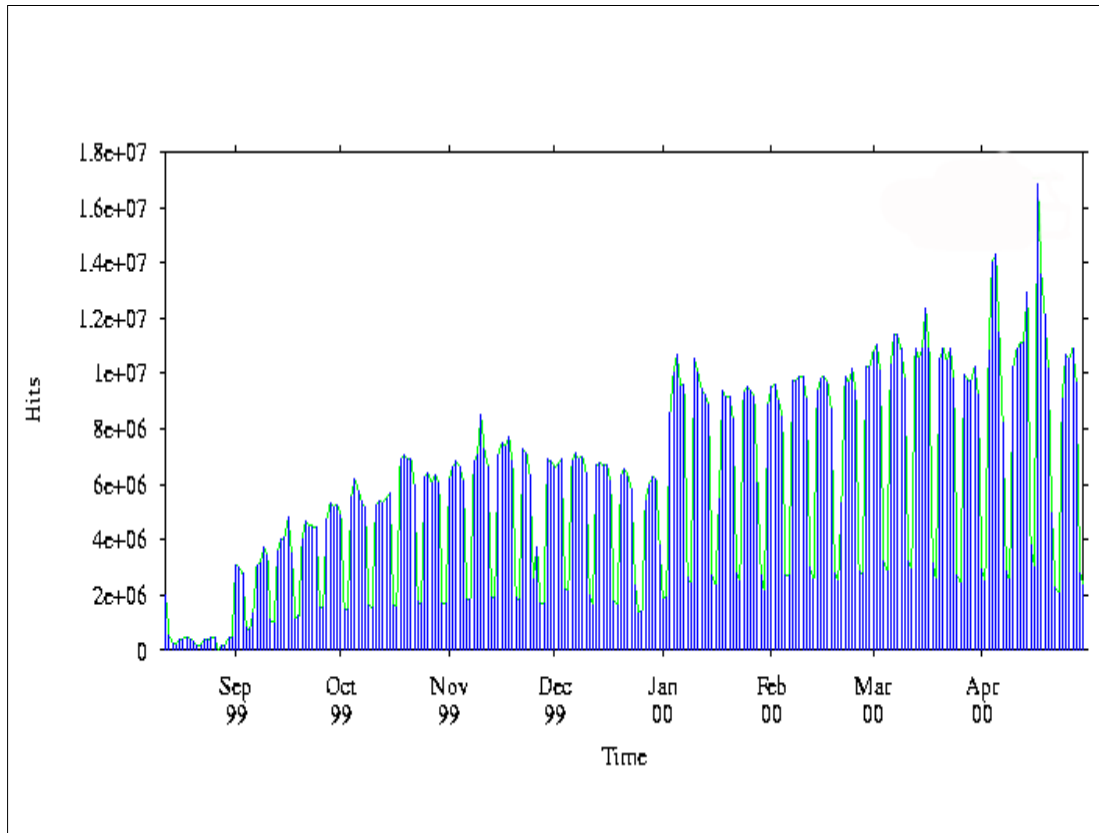


Figure 6-6 Hit rates over six months for a financial site

These examples demonstrate that it is possible to monitor your site and detect trends from which you can plan for the future and meet your business objectives. Your site will have peaks and valleys. You can measure them. You can reasonably predict when your peaks will occur and you can position the resources you need to handle the demand and serve your customers (and bring them back!).

Your trend data should suggest whether and when additional site components are needed. Powerful new servers have options, as well, that can generate capacity based on predicted workload. IBM can help you clarify which components match your particular requirements and objectives. See the Planning for Growth paper to learn about our capacity planning methodology and the HVWS Simulator for WebSphere.

Some performance management scenarios

This appendix contains three brief scenarios that are based on real events and demonstrate the principles of our methodology for managing performance.

CIO

When reviewing his schedule for the upcoming week, the CIO notes a midweek meeting with the marketing department, a Tuesday working lunch with his colleague from Finance, and the monthly CEO staff meeting on Thursday. He works with his assistant to be sure he takes appropriate information to each meeting.

On Tuesday he will take the latest reports showing costs, projected capacity over the next year, and likely capital spending. Figure 6-7 shows, at a high level, the cost per transaction

and the cost breakdown by tier. Figure 6-8 illustrates the expected growth in the number of users and transactions. These expectations were jointly reached with the marketing group. The CIO will show his Finance colleague how the increase in workload drives a needed increase in capacity and, thus, capital spending for next year. He points out that operations is working closely with application development to examine costs. They have identified where improvements can be made in the application and have projected the cost savings in terms of cost per transaction and reduced capital spending. He uses the cost savings chart in, Figure 6-9, to show how the proposed improvements will reduce the cost per transaction more effectively than the in-plan improvements.

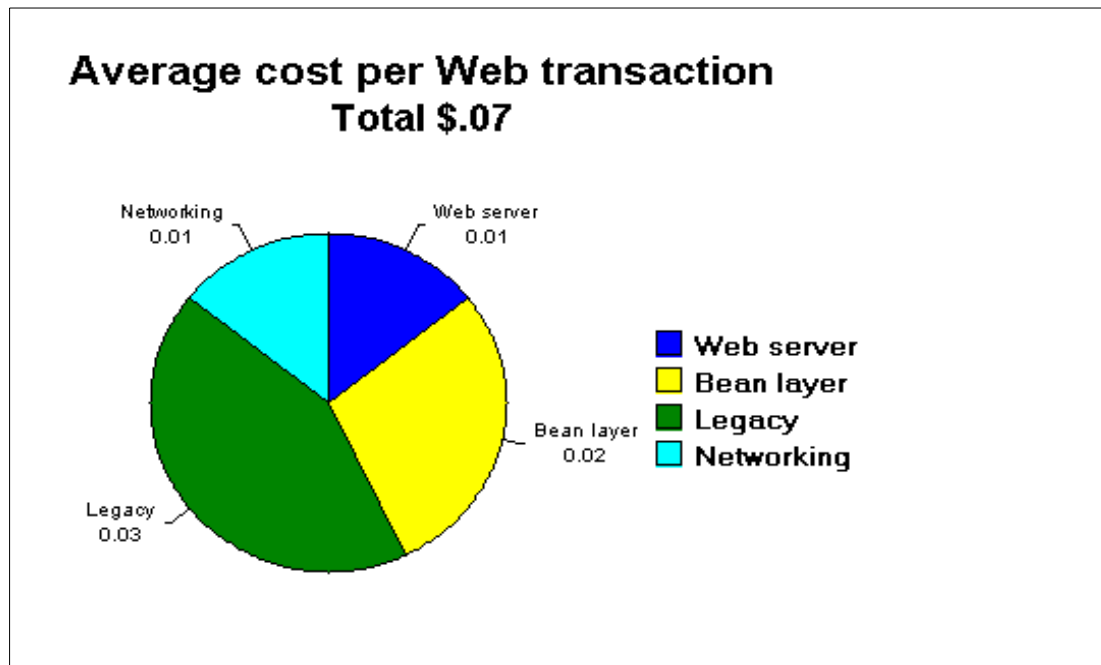


Figure 6-7 Average cost per Web transaction

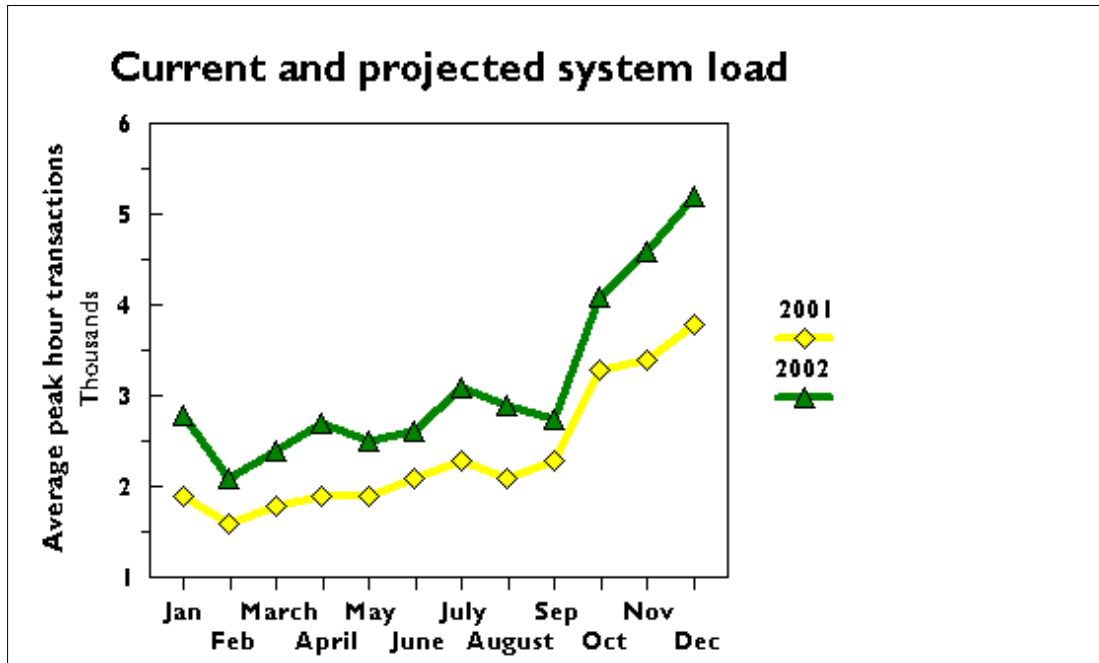


Figure 6-8 Current and projected system load

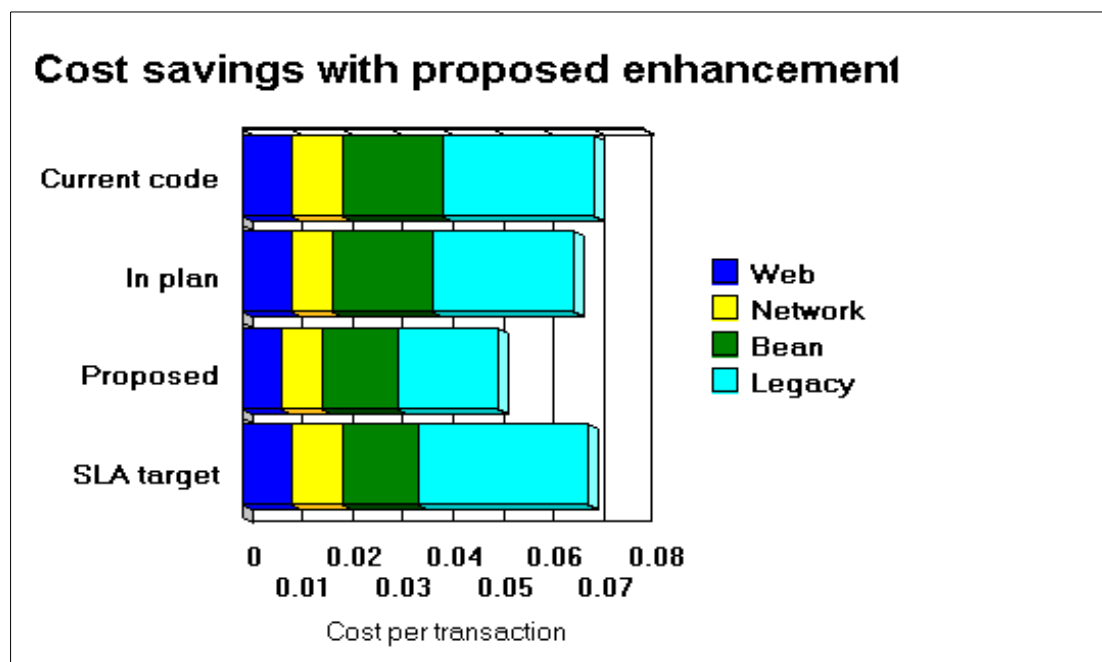


Figure 6-9 Cost savings with proposed enhancement

The CIO asks Finance to support him in prioritizing these changes in the development plan over other candidate items from other departments.

At the marketing meeting he brings the charts that report system availability, response time, transaction rates, and an analysis of consumer navigation experience. Marketing is concerned about an upcoming promotion. They expect that it will drive traffic to new highs and worry that the system will slow down. Having anticipated this line of discussion the CIO brings out charts showing the current peak demand on the system and the amount of

available overhead. He is able to demonstrate that the system has the headroom to handle up to a 30% increase in workload while still maintaining current response times during peak hours. His colleagues in marketing are pleased to see that IT has anticipated the effects of the ad campaign and are satisfied that the system will be able to contain the burst of traffic.

Finally, our CIO prepares for the CEO monthly staff meeting. Each major function is expected to present a short highlight report on the current and upcoming months. The CIO will show charts that illustrate system availability, response time and costs vs. targets. He will then discuss upcoming events, like the marketing campaign, and his plans to support them. He expects the presentation to go well because he is confident that the system is providing him the proper information to support his role.

Content problems

Last week, marketing, sales, development, and IT proudly deployed a new application that not only significantly enhanced the function of the e-business site, but also dramatically improved the look and feel of the site for the consumer.

After just a few days, however, IT noted that the IBM Tivoli Monitoring for Transaction Performance tool was producing alerts that indicated that nearly all pages were slowing down and response time was approaching the maximum allowed by the service level agreement. Using the IBM Tivoli Web Site Analyzer to examine site traffic patterns, IT observed that the site slowed down in proportion to the number of new visitors and customers. All pages were affected, indicating the problem was systemic.

IT contacted Development to review the new content. Development remained puzzled, as they had tested the new pages thoroughly before migrating them into production.

The application owner convened the performance team. One member was detailed to examine page performance using the WebSphere Studio Page Detailer. He reported: "Page Detailer shows that socket connect and SSL connect times are fine. This would seem to absolve the network, firewalls, routers, and TCP/IP layers. It also shows that transactions are processing well within criteria, so there doesn't seem to be a problem with that part of the system. However, Page Detailer does show that static content (such as GIFs) slowed down dramatically after the new application was deployed."

Armed with this information the team quickly identified the Web server as the likely problem area since it is responsible for serving up static content. As this shop was using Netscape Enterprise Server they asked for a PerfDump to be executed. PerfDump reports on the internal performance of the server. Within minutes they were able to examine the output and determine that the cache hit ratio for static content had degraded. Clearly the addition of the new application had added too much new static content to be served and the Web server cache was now too small to efficiently manage the new total. A quick look at the operating system input/output statistics using VMSTAT confirmed that real I/O had jumped dramatically within a day or so of the new application roll out.

IT was able to modify the cache size setting in the Web server and deploy the change at the next scheduled maintenance period.

Bottleneck

The e-business site was launched last month, just in time for the TV ad campaign. To date the site is successful. Traffic is growing as predicted, sales are strong, and complaints have been quite low. However, in the past few days, the application seems to have hit a bottleneck. The number of transactions has plateaued, while the response time per page has jumped dramatically.

IT employs the IBM Tivoli Monitoring for Transaction Performance tool to examine the site. They determine that only transaction pages have slowed down; the number of transactions (sales, etc.) continues to rise, while the number successfully processed is stagnant. Customers are complaining to the help desk and by e-mail about the slow response times. Analysis of the access logs produced by IBM Tivoli Web Site Analyzer confirms that many customers are leaving the site without waiting for their business to complete. Later they complain about not knowing if their business was successfully processed. A transaction in doubt is the worst possible customer problem, one that can destroy confidence in the site and the enterprise.

It's apparent there is a problem in the transaction processing. IT still checks out the Web server to eliminate it as a component of the problem. Next the team extracts the overall response times for transactions (from the Tivoli Web management solution) and uses the IBM Tivoli Performance Viewer to obtain the average elapsed times for the servlet and bean during the slowdown. Rapid subtractions demonstrate that the increased load extended the execution time of the bean. In fact, when a specific transaction rate is reached, the application can't process any more transactions in the bean layer. Additional requests exacerbate the problem in that the transaction rate remains fixed but response times become nonlinear as incoming transactions queue up waiting for the bean.

Performance Viewer at the bean engines also showed that the application server threads were busy processing requests while VMSTAT showed the CPU was less than 50% busy with no I/O or page wait. Believing that the bottleneck was found, the team recommended that additional threads be assigned to the pool so that the bean could process more concurrent requests.

Tools for monitoring performance

This section introduces some tools available to monitor Web site performance.

IBM Tivoli Monitoring for Web Infrastructure

IBM Tivoli Monitoring for Web Infrastructure is a critical tool to help ensure the optimal performance and availability of both application servers and the associated Web servers that feed them. It provides a single point of control to enable IT organizations to understand the health of the key elements of a Web-based environment. It allows administrators to quickly identify problems, alert appropriate personnel as required, and offer a means for automated problem correction. In addition, Tivoli Monitoring for Web Infrastructure provides a real-time view of performance health and feeds a common data warehouse for historical reporting and analysis. Ultimately this tool increases the effectiveness of an IT organization and helps ensure optimal performance and availability of critical Web infrastructure.

IBM Tivoli Performance Viewer

The IBM Tivoli Performance Viewer can be used with operating system tools such as vmstat to monitor a number of performance measures related to the application server. These metrics are classified into enterprise Java beans (EJBs), ORB thread pool, system runtime resources, database connection pool, and servlets. Performance Viewer is available for all WebSphere Application Server platforms.

Performance Viewer on EJB. Performance Viewer monitors execution of your EJBs at three levels: server, EJB container, and individual EJB. The table below summarizes the statistics provided as listed in Table 6-1.

Table 6-1 EJB statistics

Statistic	Stateless Session Beans	Stateful Session Beans	Entity Beans
Instantiate	Yes	Yes	Yes
Destroy	Yes	Yes	Yes
Requests	Yes	Yes	Yes
Requests per second	Yes	Yes	Yes
Execution time	Yes	Yes	Yes
Live beans (pooled and active)	Yes	Yes	Yes
Creates		Yes	Yes
Removes		Yes	Yes
Activation		Yes	Yes
Passivation		Yes	Yes
Loads			Yes
Stores			Yes

Performance Viewer on servlet. Performance Viewer monitors execution of servlets at three levels: servlet engine, Web application, and individual servlet. It monitors and collects cumulative metrics at servlet engine levels and provides an analysis of the metrics at the Web application and for individual servlets. Metrics collected include requests per second, average response time, and number of concurrent requests.

Performance Viewer on system resources. Performance Viewer monitors system resources consumed by the Java virtual machines (JVM). It collects and reports such JVM metrics as total memory and amount of memory used/available.

IBM Tivoli Web Site Analyzer

The IBM Tivoli Web Site Analyzer measures Web site traffic. Site Analyzer provides detailed analysis of Web content integrity, site performance, usage statistics, and a report writing feature to build reports from the content integrity and usage statistics. Table 6-2 summarizes the functions of each major feature of the Site Analyzer.

Table 6-2 Site analyzer functions

Feature	Functionality
Content and Site Structure analysis	<ul style="list-style-type: none"> ► Identify most popular (or least popular) page resources ► Detects unavailable resources such as broken links or missing files ► Content with excessive load time ► Web page standards and legal compliance
Usage analysis	<ul style="list-style-type: none"> ► Who accessed the site -- are they a first-time visitor, repeat visitor, or belong to a category of visitors ► Where they originated from, who referred them, length of visit ► How they navigated the site, what they did, where they exited

Feature	Functionality
Visualization and reports	<ul style="list-style-type: none"> ▶ Allow users to view site structure and quickly locate pages with problems via color schemes and icons ▶ On-demand searching for certain page attributes ▶ Provide predefined reports that are fully customizable
Fully distributed Web-based configuration	<ul style="list-style-type: none"> ▶ Extraction, transformation, and load process transforms raw data into valuable information, which is stored in a open, star database schema ▶ Client interface provides administration, visualization and report-generation functions

IBM Tivoli Monitoring for Transaction Performance

The IBM Tivoli Monitoring for Transaction Performance tool helps maintain the availability and performance of e-business systems. Transaction performance in the Web environment is maintained by measuring customer response times, executing prerecorded transactions at regular intervals, and scanning your site for potential problems. It helps you benchmark and improve Web quality of experience, giving you an understanding of your Web performance from a user point of view. For enterprise applications key performance metrics are collected and analyzed to ensure your internal customers are receiving good performance. By providing a Web to enterprise solution a complete end-to-end view of transaction performance is delivered.

Providing an end-to-end view of performance requires different types of monitors that are specialized for the different requirements of the Web and enterprise environments. To address these unique environments, Tivoli Monitoring for Transaction Performance includes these components:

Quality of Service Monitor. Captures performance data from customer transactions without using invasive server plug-ins or client agent software. When a preset threshold is exceeded it generates an alert in the form of an e-mail, a page, a Simple Network Management Protocol trap, or an event sent to Tivoli Enterprise Console®. Individual Web addresses can be easily monitored to insure key Web site links are performing well.

Synthetic Transaction Investigator. Executes prerecorded transactions from multiple locations on your site to help deliver the availability of key services and benchmark response times. Synthetic Transaction Investigator provides a browser-based record and playback function and supports a wide range of dynamic content. Alerts are generated when transactions exceed performance thresholds or fail to complete.

Enterprise Application Performance Monitor. Includes a drill-down capability in the components of key transactions, helping you pinpoint the bottlenecks of a slow transaction. You can monitor the response time of components in a multi-tier application— through instrumentation or simulation. Both methods use common services to record the response times and check them against thresholds on a Tivoli endpoint.

Site Investigator. Scans your Web site for page-size violations, HTTP errors (including broken links), pages that exceed user-defined limits and content inconsistent with corporate policy. Site Investigator generates alerts when it discovers pages that violate any of your specified constraints.

AIX performance tools

A variety of AIX tools is available to first identify and understand the work load, and then to help set up an environment that approximates the ideal execution environment for the work. Table 6-3 summarizes the AIX monitoring tools.

Table 6-3 AIX monitoring tools

Tasks	Tools	Metrics
AIX monitoring	AIX tools Perfagent tools Sample tools Adapter tools Switch tools	
Managing memory resources	vmstat sar lsps ps svmon	
Managing CPU resources	vmstat sar time cpu_state	
Managing network resources	netstat	
Netscape monitoring	Perfdump	Cache hit ratios, memory, threads
Site Investigator	IBM Tivoli Monitoring for Transaction Performance	Content Response time Availability
Quality of Service	IBM Tivoli Monitoring for Transaction Performance	Content Response time Availability
Synthetic Transaction	IBM Tivoli Monitoring for Transaction Performance	Content Response time Availability
Analyze data/generate reports	IBM Tivoli Web Site Analyzer	Site traffic analysis

Summary

Managing the performance of a high-volume Web site is challenging, exciting, and possible. Following a methodology such as the one presented in this chapter will help guide you and your team toward tasks they can understand and goals they can achieve. The success of your company's e-business depends on the tools and techniques your IT team chooses. There are many available, and more are coming, as well as capacity-on-demand options from IBM's powerful server family that set the stage for self-managing IT infrastructures. As always, their use succeeds best in the context of a process.

The 'best practices' methodology for managing a high-volume Web site includes developing an end-to-end perspective of the site and following these familiar steps:

- ▶ Establish objectives
- ▶ Monitor and measure the site
- ▶ Analyze and tune components
- ▶ Predict and plan for the future

Using this methodology, your IT team can help your company meet the revenue and customer satisfaction objectives of its e-business and enjoy improved IT performance management benefits, such as:

- ▶ Proper reporting of quality of service metrics
- ▶ Interactive and historical data on end-to-end performance
- ▶ Rapid identification of the problem source
- ▶ Improved support of business goals
- ▶ Understanding and control of transaction costs
- ▶ World class customer support and satisfaction

IBM's experience with high-volume Web sites has yielded valuable information and revealed the methodologies and tools needed for a successful e-business site. The HVWS team can help you be on your way to just such a successful site.

References

The WebSphere Application Server Performance Web site provides access to helpful performance reports, tools, and downloads at:

<http://www.ibm.com/software/webservers/appserv/performance.html>

See all the IBM HVWS papers at:

<http://www.ibm.com/websphere/developer/zones/hvws>

See all IBM Redbooks on WebSphere performance information at:

<http://www.ibm.com/redbooks>

For the latest on the IBM Tivoli performance and availability tools see:

<http://www.tivoli.com/products/solutions/availability/news.html>

The IBM WebSphere software platform for e-business includes edge servers, Web application servers, development and deployment tools, and Web applications. Find out more at the WebSphere Developer Domain at:

<http://www.ibm.com/websphere/developer>

To find out more information about the IBM WebSphere Commerce Suite, used by customers who run large-scale online shopping sites that we have studied, see:

<http://www.ibm.com/software/webservers/commerce>

To find out more information about the software used by trading sites we've studied using WebSphere Application Server, see:

<http://www.ibm.com/software/webservers/appserver>

For more information about the software used by trading sites we've studied using, WebSphere MQSeries, see:

<http://www.ibm.com/software/software/ts/mqseries>

Download a demo version of PageDetailer, the tool in WebSphere Studio that measures in detail every element in a page download to assist in performance analysis and optimization at:

<http://www.ibm.com/software/webservers/studio/download.html>



Charles Schwab puts growth plan to the test

At the forefront of online financial dealing, Charles Schwab provides high volume trading and tailored, up-to-the-minute financial information. Having developed various separate e-business channels during the early days of the Internet, their runaway success made it clear the next step was to give customers a single view of their different dealings with the company. The current infrastructure was serving its purpose well but, with the number of customer transactions rocketing skyward, Schwab needed to establish an architecture for the future that could cope with soaring growth, give greater flexibility and economy, and provide state-of-the-art customer service and satisfaction.

After thorough investigation, the obvious choice to meet their business goals was a Java[®]-based architecture built on the IBM[®] WebSphere[®] Software Platform for e-business. Having narrowed the field to this combination, Wilfried Kruse, Vice President of Architecture for Electronic Brokerage at Schwab, put up one more hurdle before the final mission-critical decision was made: "I want proof that Java/WebSphere can provide at least the same level of scalability and robustness as our current CGI (Common Gateway Interface) implementation". He looked to IBM to provide that proof.

IBM's High Volume Web Sites organization performed an extensive benchmark, which resulted in all requirements met and many exceeded, providing validation of the chosen architecture for future Schwab.com trading.

Schwab's e-business today

The Charles Schwab trading site - one of the best known in the finance industry - is just one of a number of electronic financial product channels that has evolved in response to market demand. Schwab's presence on the Web is well established and growing strongly. Today that strong growth is manageable and containable using existing technologies but to meet future demand Schwab looked for a single architecture to:

- ▶ Provide an integrated view of the different channels to customers
- ▶ Exploit the opportunities that e-business has opened up by streamlining future development
- ▶ Separate business and presentation logic to accommodate changes more efficiently
- ▶ Reduce costs

Schwab was looking for integration without losing scalability or dependability, and after careful research decided Java and IBM WebSphere Application Server looked like the answer to the new cross-channel application architecture. Java, as a leading edge technology, would attract programming skills to Schwab, and the combination of Java and WebSphere would provide the capacity for future growth while reducing the cost profile.

The proposed new architecture was code-named "Barista" (a barista is a person who serves coffee, in this case Java), and as Wilfried Kruse, VP of Architecture for Electronic Brokerage points out with tongue in cheek, "once a project has a code name it's taken seriously". To obtain proof of viability for large-scale deployment using the Barista architecture, he partnered with IBM to demonstrate that Java and WebSphere together perform as well or better than his current Web trading infrastructure.

Today's busy 24x7 Charles Schwab Web site runs on approximately 700 RS/6000s with four CPUs each, serving over 90,000 concurrent end user sessions. The majority of the current Web applications are based on C/CGI technology, which exhibits good linear horizontal scalability - adding new systems to the configuration results in a predictable increase in throughput on the Web servers.

Making sure the new architecture measures up

Wilfried Kruse, recognizing the mutual benefits, invited IBM to demonstrate that the new Barista architecture was mature enough to handle his future mission-critical workload and provide, at least, the scalability and robustness of the current C/CGI implementation. "We partnered with IBM so they could help us get the best from their technology and we could help them best utilize their technology in our market."

In response, the joint Schwab and IBM team set up the Barista High Volume Web Site Benchmark environment with four major goals:

1. Administration: To demonstrate that IBM's implementation of Java and IBM WebSphere Application Server can be installed and administered economically in a large scale, multi-node environment.
2. Stability: To demonstrate that the Java and WebSphere Application Server technologies are robust and stable, by running the environment for 48 consecutive hours without failure or performance degradation.
3. Scalability: To demonstrate that the Java and WebSphere Application Server technologies can scale at least as well as, or better than, Schwab's existing C/CGI technology, by varying the workload and number of nodes used in testing.
4. Performance: To demonstrate the Java and WebSphere technologies perform at least as well as, or better than, the current Schwab CGI technologies in comparable circumstances.

For Wilfried Kruse the key business goal was "whether the architecture and the technology could support Schwab volumes. Could we achieve comparable performance and scalability without incurring extra deployment costs? Our existing systems have known scalability characteristics and we wanted the future systems to be at least as good."

IBM benchmark

To accomplish these benchmark goals, a large-scale system environment was set up at IBM's SP Benchmark Center in Poughkeepsie representing approximately one tenth of the scale of Schwab's existing production systems. It consisted of 64 R/S6000 4-way SMP Web servers running Netscape Enterprise Server and WebSphere Application Server software, 12 additional systems emulating the corporate server functions, and several other systems driving the application workload by distributing requests across the Web servers. The administration and operational procedures were developed and enhanced during the benchmarking.

Schwab selected a critical aspect of their online trading system for the benchmark: presenting account information to customers, and a servlet was created to perform that function. For the test, five accounts were defined that differed mainly in the number of positions that the account held, resulting in varying amounts of information being delivered. The corporate server functions for the Schwab application were emulated in the test, because they use proprietary interfaces.

The results achieved for the benchmark were:

1. Administration: A Java and WebSphere installation and an efficient administration process were developed and successfully implemented for a 64-node SP configuration.
2. Stability: The stability of the large-scale WebSphere configuration was demonstrated by a soak test in which the 64-node WebSphere system ran for 48 consecutive hours without failure or errors in either the hardware or software. Performance data collected over 12 consecutive hours showed stable throughput and CPU utilization.
3. Scalability: Scalability was demonstrated in configurations up to 64 nodes by showing almost linear throughput (page views/second against number of nodes) at both low and high levels of workload.
4. Performance: The Java and WebSphere technologies performed at 300% of the stated goal. The current C/CGI technology provides approximately 1.3 page views/second/node at 37 % CPU utilization; IBM Java and WebSphere technologies achieved approximately 3.8 page views/second/node at 30 - 40 % CPU utilization. At higher CPU utilization (> 90%), 564 page views/second were attained with an average response time of less than two seconds using the 64-node configuration as shown in Figure 7-1. Assuming a configuration comparable with Schwab's installation in February 2000, this level of performance will support 100,000 concurrent end-user sessions during "market storms" (periods with very high volumes of requests).

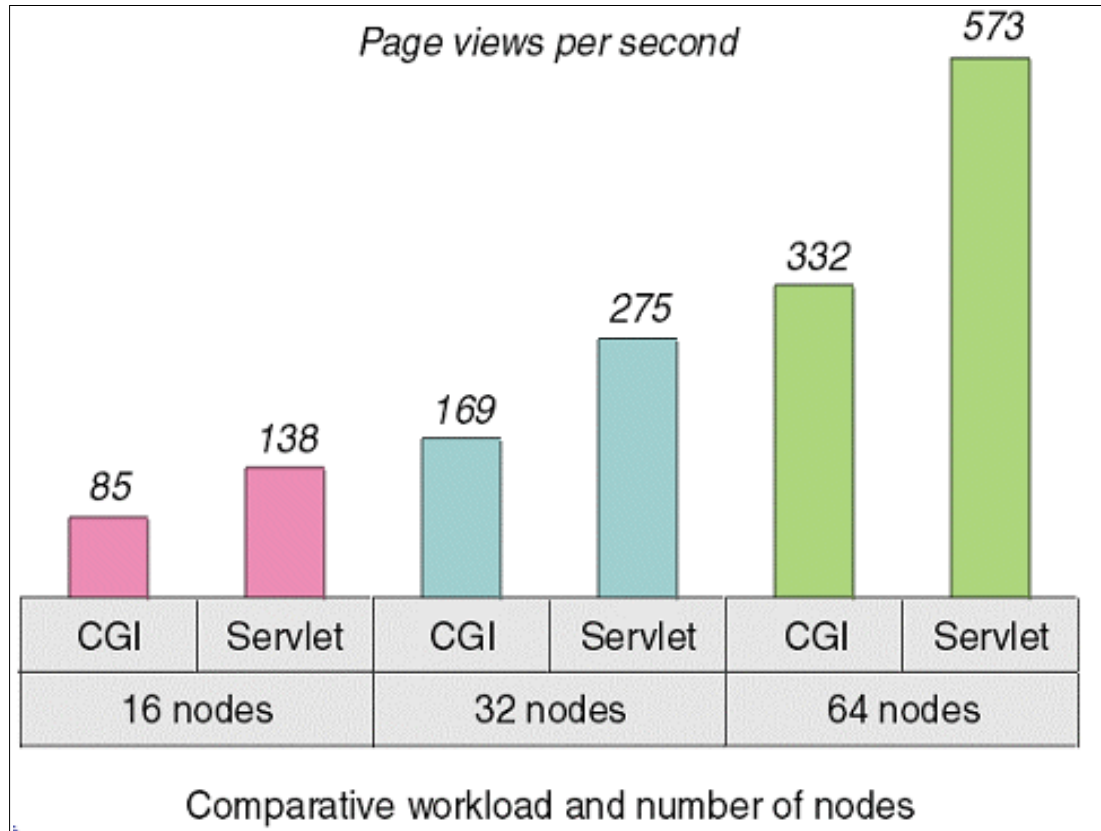


Figure 7-1 Page views per second

These results met or exceeded the criteria set by Wilfried Kruse and his team at Schwab. Not only does Barista, backed by IBM Java and WebSphere technologies, ease development, simplify the separation of business and presentation logic, reduce development costs, and provide an integrated view of the different channels to customers, but its benchmark results fully met the criteria for handling high volume mission critical workload.

Schwab's response to the benchmark results

In summary, the results achieved in the Barista High Volume Web Site Benchmark adequately demonstrated the suitability of the Schwab Barista architecture - and the IBM Java/WebSphere technologies - for high volume application use at Schwab.com. When asked about the outcome of the benchmark project, Wilfried Kruse replies, "Performance was comparable, so that was good. The stability was good, which we'd expect from IBM. But throughput was much better, by a factor of three, so that was really good because it means we can utilize the new technology, keep the existing hardware, and handle higher volumes."

"... throughput was much better, by a factor of three ..."

He also comments on having to curb the enthusiasm of Schwab programmers for the new technology until the benchmark was completed. "Once the results came out and were positive, it was like a floodgate opened. In the past some developers were interested in building their own product architectures, but now they've seen the new architecture is proven, and it's simple enough and yet sophisticated enough to handle our business, they're saying 'let me use it', which is a really good change."

"Initially we'll be writing a lot of infrastructure that already exists - reinventing the same wheels in the Java space - so there won't be savings straight away, but once we have a critical mass of the new infrastructure and products deployed in this new environment, new development will be much more productive than in the current C/CGI environment."

"Once we have a critical mass of the new infrastructure and products deployed in this new environment, new development will be much more productive than in the current C/CGI environment."

And what benefits will Schwab's customers see? "They'll see more consistent products, and more coherent product behavior," says Kruse "because there will be a more integrated front-end presentation layer across the different financial product channels." Other benefits are that "Schwab can roll out new functions in smaller increments much faster, our product release procedure will be more efficient. And quality will improve because we'll get a lot more reusable code - that means code that's already tested will be used in new ways, it won't need to be reinvented. I expect a lot of productivity and quality gains from that."

"Our product release procedure will be more efficient and quality will improve."

And Wilfried Kruse recognizes the value of partnering with IBM, "A very encouraging effort, I'm very pleased with how it went." Follow-on work is to explore the use and scalability of EJB (Enterprise Java Beans) technology.

IBM partnership executive Keith Jones reflects on the results achieved, "This project presented a significant challenge to both the Schwab and IBM teams. The results speak to the strength of the Barista architecture and the strength of IBM Java and WebSphere technologies and the people behind them. Schwab are leading exploiters of these technologies; their leadership translates to a quantum leap for large scale e-business."

Technical benchmark details

The technical benchmark details are covered in this section.

Goals

The four major goals were:

1. Administration: To demonstrate that WebSphere can be installed and administered in an environment with a large number of nodes. Sets of nodes would be administered at the same time, with a single command to any set of nodes. This technique was verified against the 64 Web server nodes and 12 back-end nodes, and was used for all components in the setup.
2. Stability: To demonstrate that Java and WebSphere Application Server technologies are robust and stable, by running the environment for 48 consecutive hours without failure. System performance, judged by a few key performance indicators, would be constant over the period. IBM would monitor the rate at which the system processes requests to ensure that it could handle the same load throughout the period. Response times of the requests would also be monitored to make sure the CPU utilization was not increasing over the period.
3. Scalability of Servlet: To demonstrate that Java and WebSphere Application Server technologies will scale. Linear scalability is achieved if twice as many nodes can handle twice the throughput with the same CPU utilization of each node. IBM tested this by varying the number of nodes and the throughput. They also verified that as the number of

nodes increased the throughput would increase proportionally. CPU utilization and response time would also be measured.

4. Performance of CGI vs. servlet: To demonstrate that Java and WebSphere technologies perform as well as or better than the CGI technology in Schwab's current Web trading environment. The WebSphere servlet and CGI both running under Netscape in the same configuration would be compared for performance, throughput, CPU utilization, and response time.

Test scenarios

The test scenarios exercised a servlet provided by Schwab from their Barista architecture project. The servlet performs a critical aspect of the online trading application; it allows users to view their account information. The test implemented five accounts that held varying numbers of stock positions in the account portfolio. Each account returns a different size page to the Web browser.

The servlet provides business logic that supplies data to a JSP (Java Server Page), which is then returned to the user's Web browser. The servlet obtains much of the information that it displays by using a proprietary interface to a back-end server. For the purposes of this benchmark, the functions of the back-end were emulated by a C++ program and a set of sample files running on an AIX system.

For the servlet vs. CGI performance comparison, Schwab provided production CGI programs and servlets with JSPs for the comparable application function.

The servlet function contains business logic that fills in different areas on the screen. First it performs a logon, and when done performs a logoff, so that it only obtains information that this user is allowed to see. It has the ability to handle user preferences. It shows urgent notifications, retail offerings, news items, and a graph of current market status. It also shows users their specific account balances, and a list of each position they hold. In the smallest account, there are no positions held, while in the largest account there are hundreds of positions.

The servlet obtains much of the information for these sections through proprietary interfaces to back-end systems, which Schwab has developed. For the purposes of this project, the functions of the back-end were emulated by a Unix program and a set of files.

Benchmark environment

In IBM's SP Benchmark Center in Poughkeepsie NY, the setup was 64 SP Silvernodes running WebSphere Application Server Advanced Edition 3.0.2, twelve additional nodes to emulate the back-end server function, and several other machines used to drive the system (using Mercury Interactive's LoadRunner) and to distribute requests (using IBM Network Dispatcher). All these nodes were connected together using a CISCO 6509 switch. Each Web server had a single 100MB LAN connection. Each back-end server had two 100MB LAN cards, and the other machines had one or more gigabit LAN cards.

- ▶ Twenty four-way S80 and 12-way S7A machines were used to simulate the workload. These systems were selected based on the anticipated workload by the test driver.
- ▶ Four-way F50 (H70) machines were selected as network dispatcher. Their function was to dispatch and then balance the workload on 64 Websphere nodes. These are the most commonly used machines for the dispatching function
- ▶ Sixty-four SP Silvernodes (332 MHz, 4-way RS/6000 SMP nodes) were used to configure WebSphere. These nodes were used to measure the administration, performance, stability, and scalability of the system. These are the same nodes that are used in Schwab's production system.

- ▶ Twelve Winterhawk (375 MHz, 4-way RS/6000 nodes) were used to simulate the delay of the back-end database server. These nodes were based on the anticipated workload and hardware availability.
- ▶ Network topology: Fully meshed 64 fast Ethernet connections were configured, one from each Web server, to the CISCO 6509. Additional 24 fast Ethernet connections to the switch used for the Etherchannel connections to each of the 12 Echo servers (two fast Ethernet per node). Lastly a total of 10 separate Gigabit connections were in place for the two Load drivers (S80: 4 Gigabits & S70: 2 Gigabits) and the four Network Dispatchers (1 Gigabit each). Network configuration was selected based on the anticipated load.

Software configuration

The software configuration is shown in Figure 7-2 and consists of:

- ▶ IBM AIX 4.3.3
- ▶ IBM Java JDK 1.1.8
- ▶ IBM WebSphere Application Server Advanced Edition 3.02
- ▶ IBM DB2 UDB 5.2
- ▶ IBM Network Dispatcher 2.1
- ▶ Netscape Enterprise Server 3.6

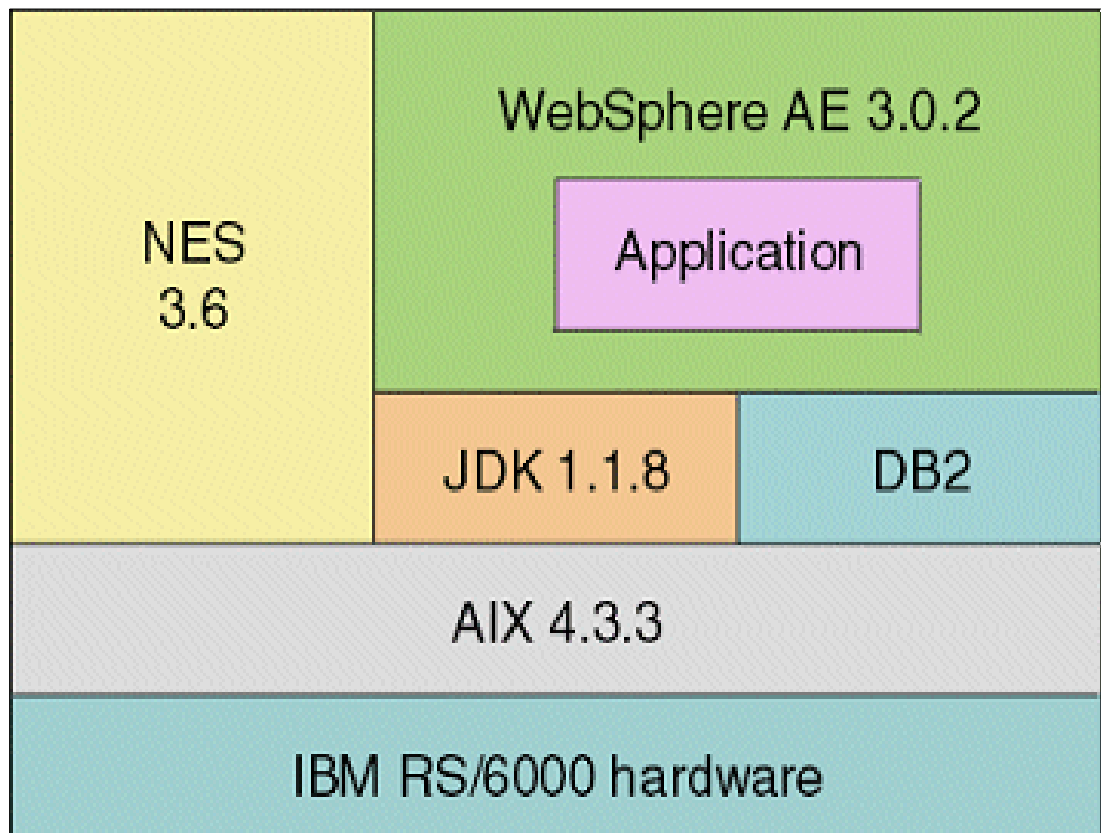


Figure 7-2 Software configuration

For more information, contact:

- ▶ Charles Schwab:

www.schwab.com

- ▶ IBM WebSphere:

ibm.com/WebSphere

- ▶ IBM Java:

ibm.com/Java.



Fine-tuning the scalability of a multi-tier architecture

Charles Schwab and Co., Inc. (member SIPC/NYSE), long a pioneer in online financial services, determined in early 2000 that a new e-business architecture for the future was needed to manage future growth, give greater flexibility and economy, and provide state-of-the-art customer service and satisfaction. Consistent with Schwab's forward thinking, they decided the clear choice to meet their business goals for the retail business was a multi-tier Java-based architecture built on the IBM® WebSphere® software platform for e-business.

IBM and the Schwab Integrated Client Technology (ICT) and Java Object Services (JOS) departments designed and developed a new architecture, code-named Barista.¹ Relying heavily on IBM's hardware and expertise, the joint team executed a set of benchmark tests to validate the architecture and determine how to balance the number of servers in each tier for optimal performance. At the beginning of Barista, Wilfried Kruse, VP of Architecture for ICT, gave IBM the challenge: "I want proof that Java/WebSphere can provide at least the same level of scalability and robustness as our current CGI (common gateway interface) implementation". The success of the first phase of Barista is described in the white paper, Charles Schwab puts growth plan to the test.

This chapter describes the subsequent work with the Barista architecture. One of the enhancements key to meeting Schwab's business goals of economic scalability and consistency across customer channels was to separate their business logic and presentation layers into separate tiers. The presentation tier was implemented using servlets and the business logic tier was implemented using Enterprise JavaBeans (EJB). Barista also implemented new technologies in the presentation and business logic tiers. In the presentation tier, servlets use eXtensible stylesheet language (XSL) and the Xalan eXtensible stylesheet language transformations (XSLT) processor. In the business logic tier, the database implementation uses an IBM DB2 database and IBM pSeries S80 technology

Barista testing demonstrated the scalability, performance, stability and manageability of the Barista architecture in a multi-tier environment. The last phase of Barista development culminated with the testing of a new home page application for Schwab's financial customers called myHome. The tests emphasized hardware/software capacity planning numbers, performance, and stability. At each phase, the test results validated Schwab's architectural

decisions. Stability, scalability, and performance results exceeded expectations. The testing demonstrated near linear scalability.

The Barista project validated Schwab's decisions to move to a multi-tier, Java-based architecture built on the IBM WebSphere software platform for e-business environment. The architecture satisfied Schwab's key business goals.

The Barista project had great significance to Schwab and to IBM. It demonstrated what is possible with the new Java technologies when implemented in the multi-tier environment. It is symbolic of what can be done and must be done as we increase our understanding of how best to implement the new technologies that will shape the future of e-business. With the advent of new standards such as Java 2, Enterprise Edition (J2EE) and Web services, there is much to do to assure effective implementations and continued e-business success.

Introducing the Barista project

The requirement to create a new e-business infrastructure for Schwab's retail electronic brokerage gave both Schwab and IBM the opportunity to apply the latest in technologies and best practices. Schwab presented the challenges of managing economic growth in a dynamic business environment, and IBM presented practices based on its accumulated experience with some of the world's largest Web sites and accumulated knowledge of what to do and what to avoid.

Both companies contributed their best skills to the team that would explore what was possible and necessary to design, validate, implement, and manage a complex new infrastructure. The Barista project proceeded in phases. Each phase concluded with tests to validate the performance and scalability of the architecture within a single domain² and across multiple domains. Testing focused on the architecture's performance and ability to scale to anticipated business volumes. As one of the largest online brokers with a need to manage growth incrementally, Schwab made scalability one of their top priorities for the Barista architecture.

The Barista architecture is based on Java and built on the IBM WebSphere software platform for e-business. In the beginning, the Barista architecture used a Java server page (JSP) for presentation logic and a servlet for business logic in a single tier³. The team extended Barista to introduce a multi-tier architecture by separating presentation and business logic layers into separate tiers. This critical step enabled Barista to:

- ▶ Segregate functions in a way that allows consistent presentation across channels and reuse of application code and data
- ▶ ·Exploit the scaling benefits inherent in multi-tier environments
- ▶ ·Create a foundation for manageable growth

Figure 8-1 shows the current topology of the Barista test environment.

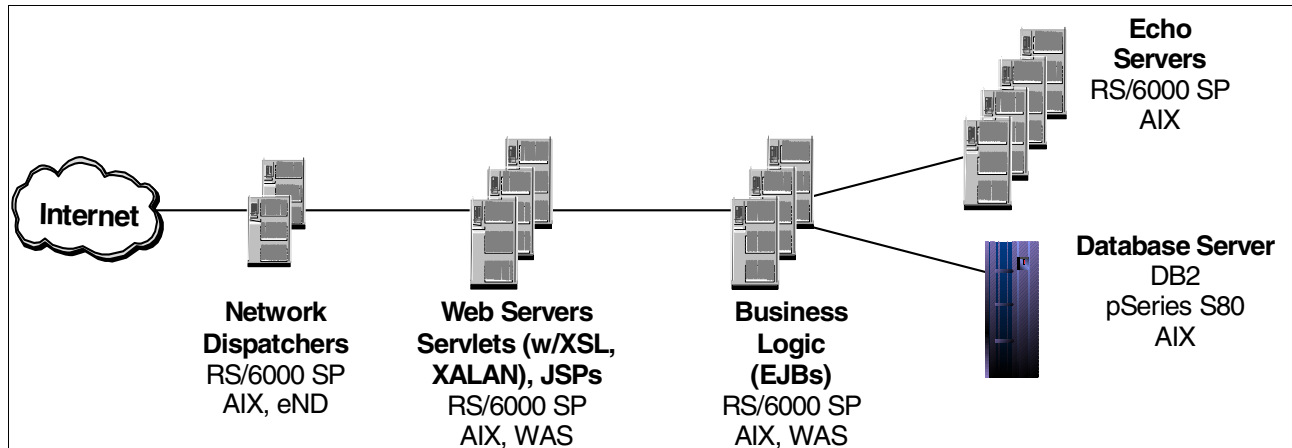


Figure 8-1 Topology of Barista test environment

Separating presentation and business logic laid the foundation for disciplined application development. The presentation tier was implemented using servlets and the business logic tier was implemented using Enterprise JavaBeans (EJBs). Finally, the team implemented new technologies in the presentation and business logic tiers. In the presentation tier, servlets use XSL and the Xalan XSLT processor. In the business logic tier, the database implementation uses an IBM DB2 database and IBM pSeries S80 technology. The tests used echo servers to simulate the transaction system.

Testing Barista

Testing focused on Barista's performance and ability to scale to anticipated business volumes. Workload balancing was used to determine the number of servers in the presentation and business logic tiers that produced the best performance. This is an iterative process that must be followed to insure the performance and scalability of a multi-tier solution and attain the most efficient use of resources in each tier. Typically, an organization will have to execute tests of this nature several times to determine how many servers in each tier yield the best performance.

The Barista tests consisted of variations on two scenarios. The variations involved increasing the number of servers in a domain in the single domain tests and varying the number of domains in the multiple domain tests. This section describes the scenarios and some of the key results; it concludes with a summary of the results and our conclusions.

Mercury Interactive's LoadRunner tool was used to generate the workloads. Each test driver was specified to simulate typical user scenarios accelerated to computer speed, essentially eliminating the latency of human interaction. The combined drivers (as many as 54) simulated a heavier than typical user workload, able to saturate the system more intensely than actual human users.

Barista EJB test scenario

Figure 8-2 shows the topology of the Barista EJB test.

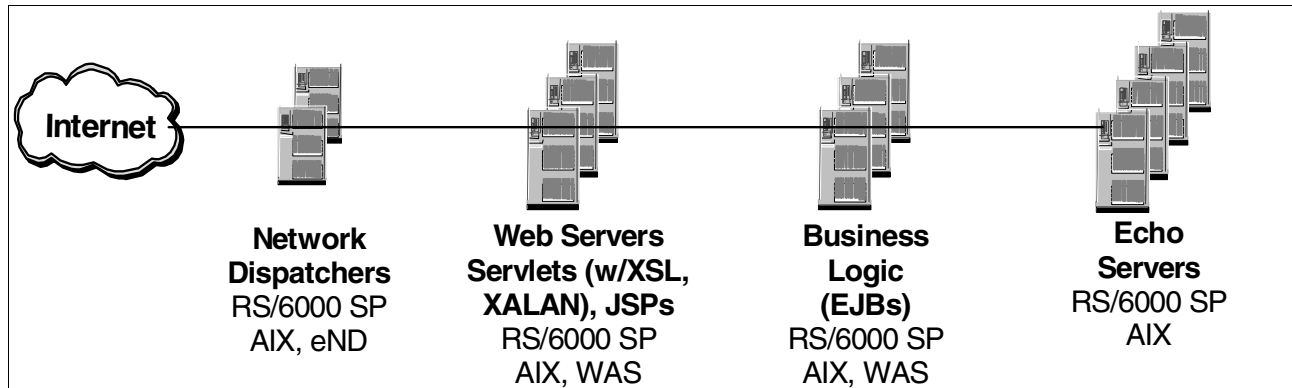


Figure 8-2 Topology for Barista EJB test

The test scenario used the Account Overview application, which displays information about a customer's account. To test varying loads, the tests accessed two sample accounts, 906 and 939. Tests using account 906 placed higher loads on the database than those using account 939. The results showed that the transaction throughput scaled linearly, while the resource consumption per transaction and response time remained relatively flat.

Figure 8-3 shows page throughput measurements in the multiple domain tests as additional domains are added. The tests used one, five, ten, and 15 domains. Each domain consisted of three servers in the presentation layer, one server in the business logic layer, a 3:1 ratio, and 54 test drivers.

This test offers a strong proof point to the inherent scalability of the WebSphere Application Server (WAS). WAS exploits caching within its architecture, and minimizes resource management overhead by pooling key system components, such as connections and EJBs.

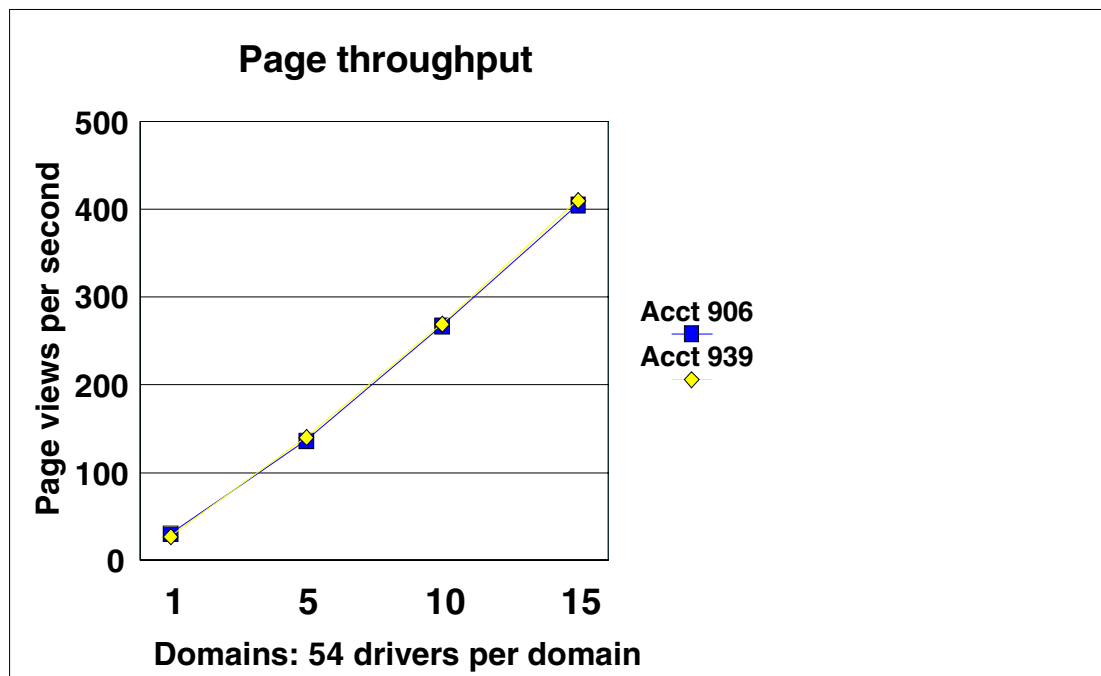


Figure 8-3 Results of the Barista EJB multidomain test scenario

WebSphere's workload management capabilities offer flexible alternatives for ensuring scalable load balancing. IBM's long-standing experience in designing scalable, balanced,

symmetric multiprocessing systems (SMPs) has also been applied in designing WebSphere, enabling near-linear scaling in a distributed, multiserver configuration.

In the single domain tests, WebSphere and the Barista application code scaled proportionally as servers were added to the domain. Note in Figure 8-4 that the largest domain -- 44 presentation logic servers, 15 business logic servers, and 15 database servers -- comprised an unusually large WebSphere domain. The facilities at IBM's HVWS Lab in Poughkeepsie were used for these tests. The transaction rate scaled linearly as servers were added to the measurement domain. The average response time remained within acceptable limits and decreased in the largest domain. CPU utilization per server remained consistent.

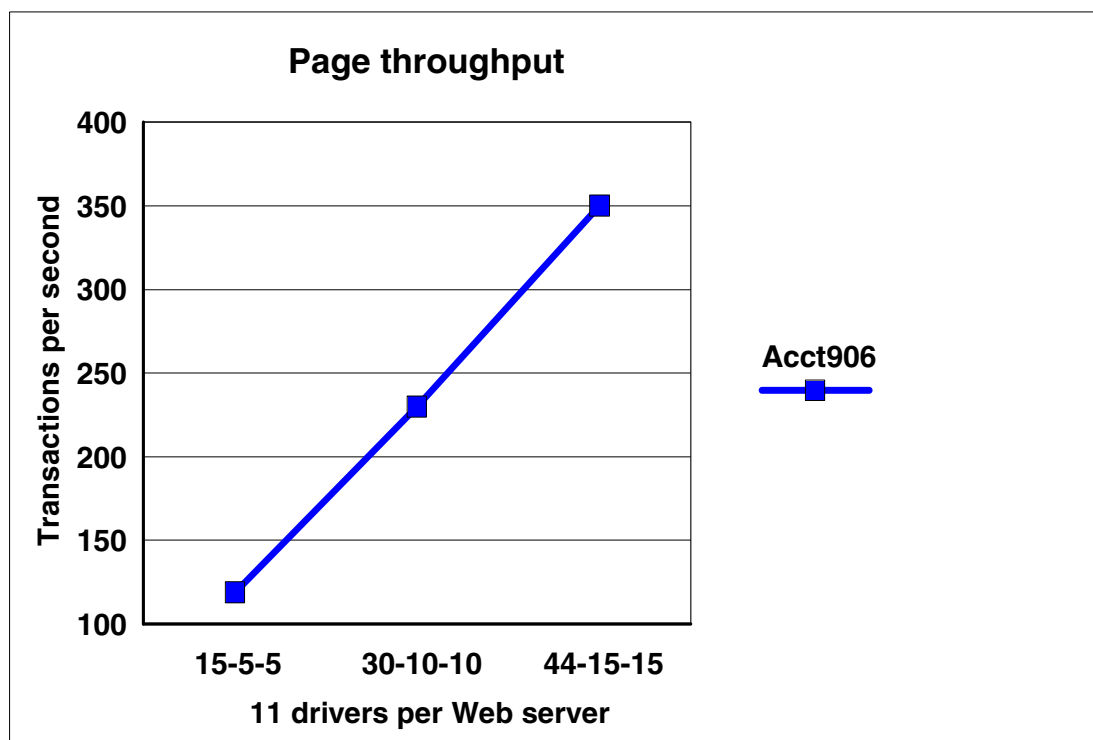


Figure 8-4 Results of the Barista EJB single domain test scenario

myHome application tests

At this point in the project, Barista implemented new technologies in the presentation and business logic tiers as shown in Figure 8-5. In the presentation tier, servlets use XSL and the Xalan XSLT processor. In the business logic tier, the database implementation uses an IBM DB2 database and IBM pSeries S80 technology. Barista testing focused on specific tests necessary to support the planned deployment of a new home page for Schwab's financial customers called myHome. The myHome application displays a home page personalized with information specific to each customer. The tests emphasized hardware/software capacity planning numbers, performance, and stability. The tests also focused on workload balancing to determine how many servers in each tier produced the best performance measurements.

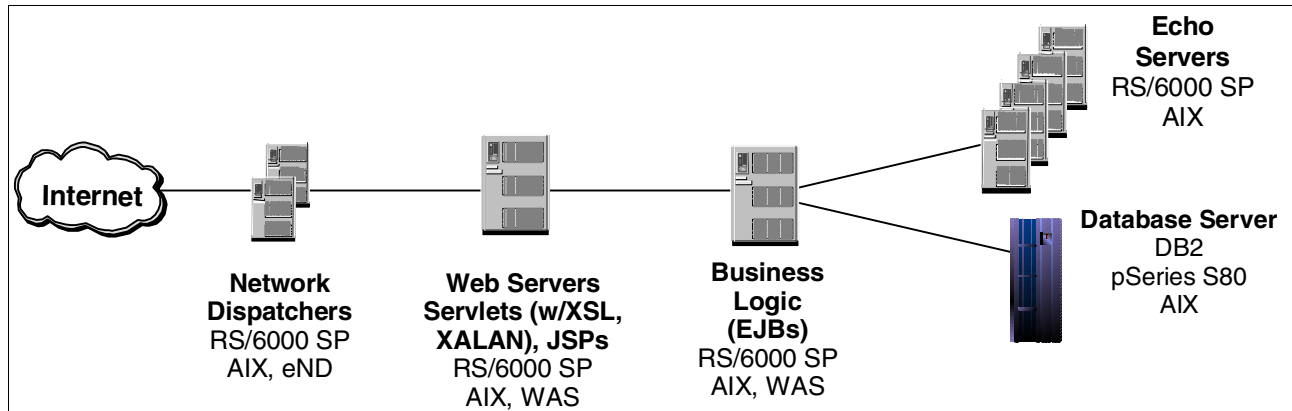


Figure 8-5 Topology for Barista test (one presentation and one business logic)

The goal of the tests run against the Barista myHome application was to measure the throughput and response time under high volume. The first tests determined the baseline throughput performance with and without security. Then the full scenario was run to determine the maximum throughput achievable with one server in the presentation layer and one server in the business logic layer.

Figure 8-6 shows the tests results, indicating that a single presentation server did not saturate the business logic layer. As load increased, the presentation server peaked at 95% busy while the business logic server was only 68% busy. Note in Figure 8-6, when the presentation server reached 95%, increasing the load did not result in significant increases in transaction rate, but did increase response time. This exemplifies the sort of nonlinear behavior that can occur when CPU utilization exceeds 90%. It was clear that to achieve optimal performance, it was necessary to add presentation servers.

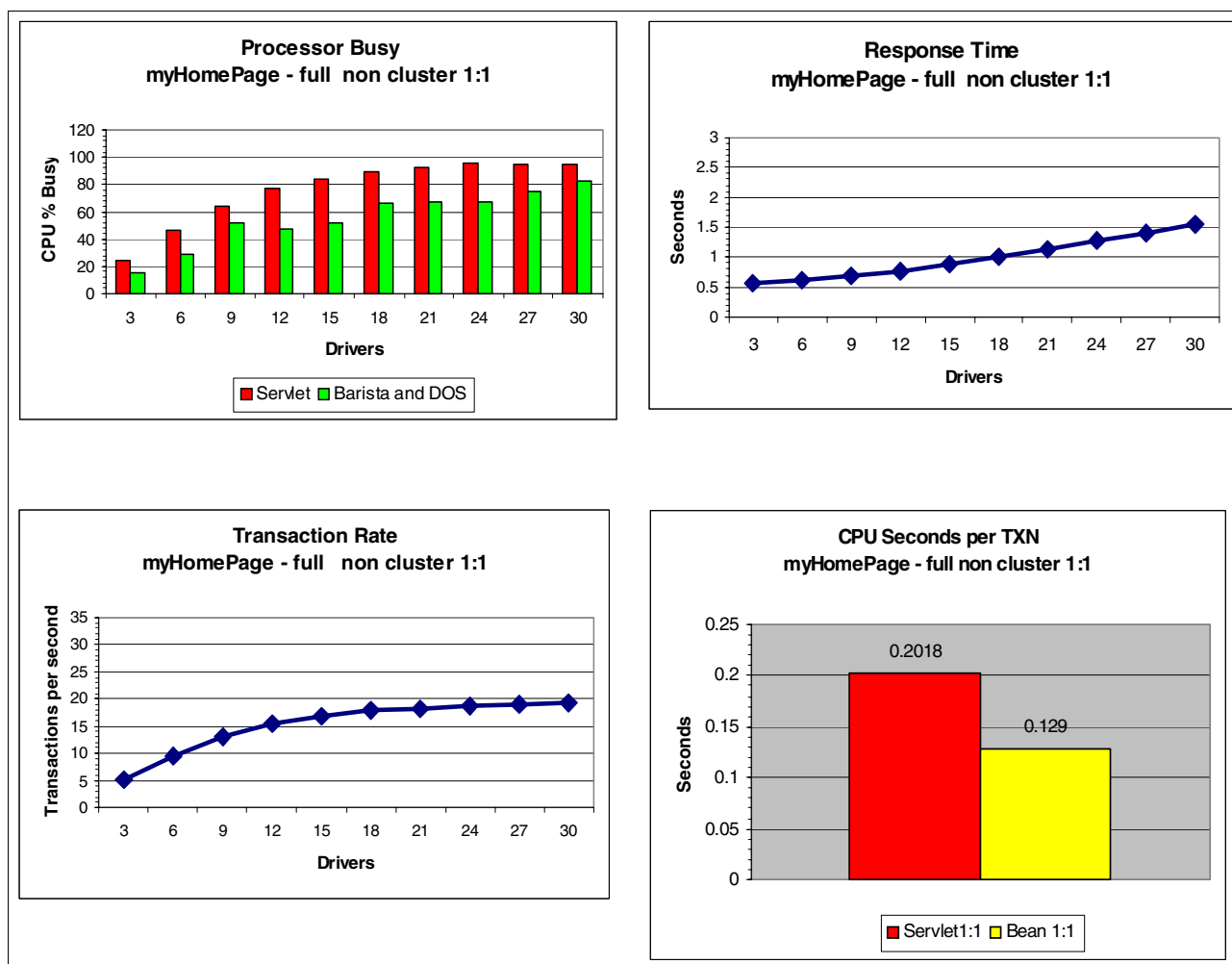


Figure 8-6 Results of Barista test (one presentation server and one business logic server)

The next test was designed to determine how the application scaled as additional presentation servers were added. As shown in Figure 8-7, the test used two presentation servers to one business logic server, a 2:1 ratio. As load increased the business logic server peaked at 93.9% busy while the presentation servers averaged only 73.9% busy.

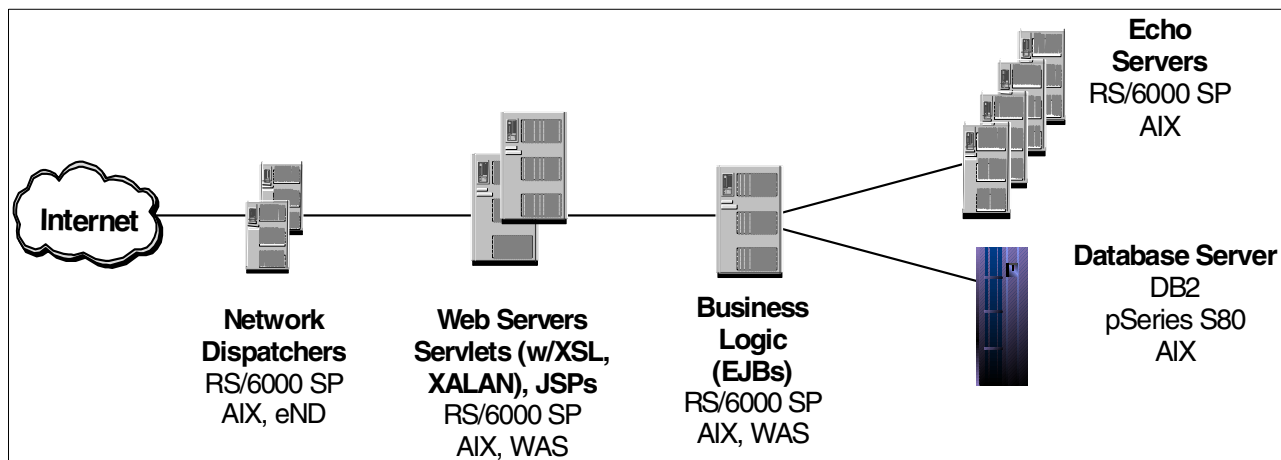


Figure 8-7 Topology for Barista test (two presentations and one business logic server)

The results shown in Figure 8-8 indicate that 42 test drivers (simulating 42 users) were able to saturate the business logic layer to 93.9% CPU utilization, while the presentation servers averaged 73.9% busy at a transaction rate of 29.4 per second with a response time of 1.4 seconds.

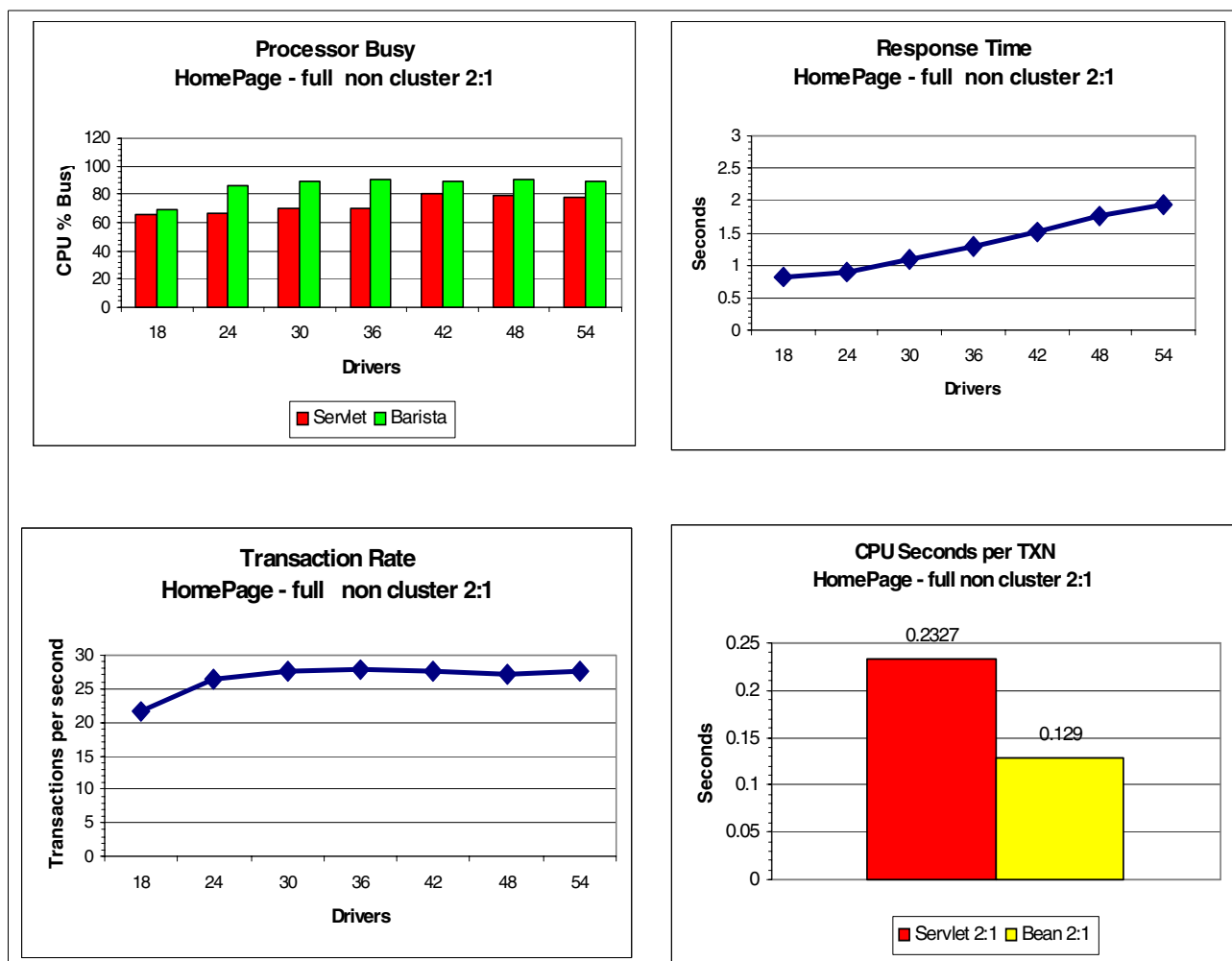


Figure 8-8 Results of Barista test (two presentation servers and one business logic server)

Here we see that by increasing the number of presentation servers, the business logic server could be fully utilized. It is also an excellent example of how changing the capacity of one tier can affect the load on an adjacent tier. Adding servers to the presentation tier resulted in increased load on the business logic tier, fully utilizing the processors in that tier. This is one step in the iterative process of workload balancing that must be followed to attain the most efficient use of resources at every tier. Typically, an organization will have to execute tests of this nature several times to determine the number of servers in each tier that yields optimum performance. In each test, the numbers of servers may be adjusted and new measurements taken. It is important to use workloads that most closely approximate real production loads. When the new configuration is in production, under real life workloads, resource utilization in each tier needs to be carefully monitored. IBM has tools with built-in workload models that can be used to estimate resource requirements.

Finally, to determine stability, the testers ran a soak test for 25 hours. The test used two presentation servers and one business logic server. The presentation servers averaged 76% busy and the business logic server averaged 93% busy at a load of 60 page views/second.

See Figure 8-9 for details. The small dip seen eight hours into the run occurred when the HTTP server executed a cleanup service routine to recycle its threads.

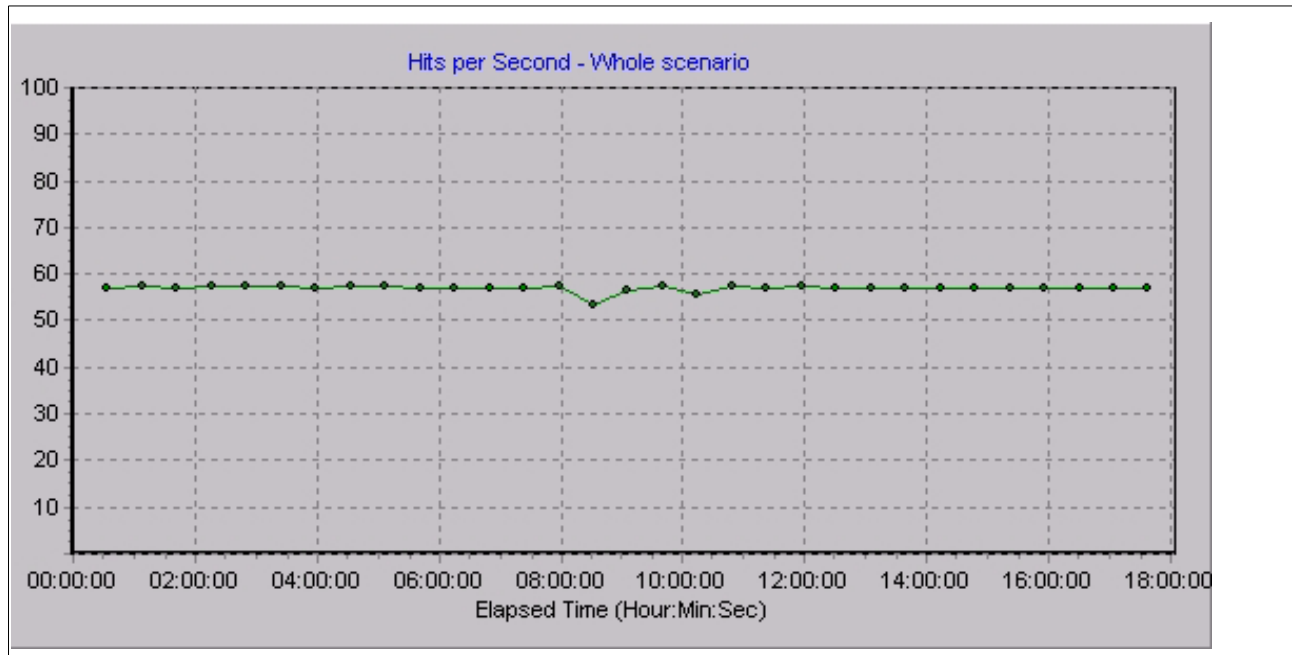


Figure 8-9 Results of Barista soak test

Summary of Barista results

Scalability: WebSphere Application Server and the Barista application code scaled in a near linear fashion throughout the load range of individual systems and from a single to many domains and also within large domains. Scaling in the business logic layer was linear as the workload increased from 18 through 54 drivers. The application and the system behaved well throughout the measured range, and response time increased evenly from 0.7 to 1.4 seconds. These results validate the soundness of the Barista architecture and inherent scalability of the WebSphere Application Server.

Workload balancing: The objective of workload balancing is to determine the number of servers in each tier that yields consistent CPU utilization across tiers, while driving high workloads without exceeding 90% CPU utilization. CPU utilization above 90% may result in nonlinear scaling. Barista testing verified a ratio of three presentation servers to one business logic servers. Schwab and IBM agreed that the Barista test workload did not represent production experience and agreed that follow-on testing with more representative workloads will be considered. During the testing of the myHome application, server ratios of 1:1 and 2:1 were tested, with 2:1 providing the best workload balance. In addition to the number of servers, factors that contribute to the performance of a multi-tier architecture include workload mix, data transferred between physical layers, number of HTTP connections per HTML page, upstream caching, and application design.

Performance: Each test obtained performance measurements. In testing the myHome application, tests were successfully run on the two presentation servers and one business logic server configuration. Maximum throughput in this environment was achieved with 42 drivers driving 29.4 transactions per second with a response time of 1.4 seconds. In the EJB test scenarios, performance was demonstrated across 15 domains with 400 transactions per second achieved, using a complex total account overview transaction. Each domain

consisted of three presentation servers and one business logic server. Response time below two seconds was achieved with CPU utilization above 80 percent. A single domain of 44 presentation servers and 15 business logic servers was constructed. This system maintained an average response time below two seconds and CPU utilization above 75 percent while delivering over 350 transactions per second.

Stability: Soak tests were used to demonstrate the stability of the infrastructure. In a 25-hour soak test, the business logic tier was over 90% busy, the system remained stable, and all results processed normally. Testers determined it was unnecessary to continue the test past 25 hours as the testing in Barista had already shown the stability of the architecture. Stability in the large-scale environment was confirmed by completing a 43-hour soak test. Transaction load averaged 140 page views per second with a zero error rate. A consistent average response time of 1.4 seconds was achieved. This test was conducted in a server ratio of 15:5:5 (presentation / business logic / echo) domain with average CPU utilization at 75% for the presentation tier and 84% in the business logic tier.

Manageability: The team developed and successfully implemented processes for installing and administering Barista in the WebSphere environment. The results indicate that the Barista and WebSphere application code can be deployed, operated, and managed successfully in a high volume environment. The domain of 44 presentation servers and 15 business logic servers is an unusually large single WebSphere domain. Complexity of system operation increases as the number of systems grows. This consideration requires attention during deployment. The High Volume Web Site team described to Schwab the methodology used to recognize bottlenecks and implement tuning. Schwab management asked that the information and methodology be shared within Schwab.

Code review: Experts from IBM reviewed the Barista architecture and code. Schwab management has requested future reviews. Motivated by interim test results, IBM analyzed Java performance using Java profiling tools to identify areas in the Schwab code that were candidates for reducing path length and effect on system resources. This work was repeated using the final test configuration to provide Schwab with the most current results. Schwab management requested access to the necessary tools and education to enable their ongoing use of the techniques.

Best practices

IBM reviewed the Barista architecture. The review identified some of the best practices used in the design, implementation, and testing of the Barista architecture.

Multi-tier architecture: Schwab designed the Barista architecture to be layered, allowing it to scale to handle growing workloads and be tuned to make the most efficient use of system resources. WebSphere Application Server has been designed to operate efficiently in multi-tier environments. It exploits caching within its architecture, and minimizes resource management overhead by pooling key system components such as connections and EJBs. Its workload management capabilities offer flexible alternatives for ensuring scalable load balancing. IBM's long-standing experience in designing scalable, balanced, symmetric multiprocessing systems (SMPs) has also been applied in designing WebSphere, enabling near-linear scaling in a distributed, multiserver configuration

Designing for scalability and planning for growth: Schwab understands the critical importance of designing for scalability and planning for growth. Following our HVWS best practices, they understand the significance of analyzing the scalability of the components of the presentation and business logic tiers. They selected appropriate scaling techniques, effectively using horizontal scaling and workload balancing. As their architecture evolved, they

reevaluated their scaling and workload balancing to ensure solutions that will scale effectively during workload peaks.

Workload balancing: Barista testing focused on determining the number of servers in each tier that yielded the best performance and scalability. The need for this testing is indicated whenever a significant change to the infrastructure is made. Some examples of such changes are additions or changes to hardware or software in anticipation of increased volumes, the addition of an application, or even the installation of an upgrade to any key software component. The techniques used will allow Schwab to accurately tune their production systems as they deploy future applications using the Barista architecture. WebSphere Application Server's workload management capabilities offer flexible alternatives for ensuring scalable load balancing.

Style sheet caching: Schwab designed the Barista architecture to use XML and XSLT. Recognizing the potential for performance problems in XML/XSLT solutions, Schwab made two good decisions that allowed them use XML and XSLT efficiently. They implemented style sheet caching and built document object model (DOM) trees (in-memory documents) rather than using XML strings. Effective caching is a powerful tool to improve both scalability and site response times. There are at least seven levels at which Web sites can exploit caching. Style sheet caching is an effective way to improve the performance of XML based solutions.

Dynamic caching: The possibility of using the dynamic caching feature in WebSphere 3.5.3 was under investigation at the end of the Barista testing. Dynamic caching allows dynamically built Web pages to be held in a cache that can be keyed by any of a number of objects, for instance, parts of the HTTP request, the user session, user id, etc. If a matching page is found in the cache, the servlet or JSP is not even invoked. Instead, the cached value is returned. Schwab continues its evaluation of dynamic caching.

Summary

The joint project between IBM and Charles Schwab succeeded not only because the business and technical objectives were met, but also because of the combined strengths that each company brought to the project. With significant foresight, Schwab contributed their challenging environment for the future of online trading and their electronic brokerage. IBM contributed skills and products that addressed those challenges. Clearly, WebSphere's broad range of features and functions, as well as its implementation using J2EE, met the challenge. The Barista project demonstrated that:

- ▶ Schwab's Barista architecture performs well and scales to anticipated business volumes.
- ▶ A multi-tier architecture is a viable and correct choice for large, growing workloads. Scalability, performance, and stability measurements met or exceeded objectives.
- ▶ WebSphere scales in a near linear fashion as workload is increased.
- ▶ WebSphere applications can be deployed, operated, and managed successfully in a high-volume environment implemented in a multi-tier architecture.
- ▶ WebSphere exploits caching within its architecture.
- ▶ WebSphere minimizes resource management overhead by pooling key system components such as connections and EJBs.
- ▶ WebSphere's workload management capabilities ensure scalable load balancing.

The IBM and Schwab teams collaborated throughout the project. IBM experts assisted Schwab in designing the architecture. IBM supplied HVWS test sites at the Silicon Valley Lab and the Poughkeepsie lab. Technical staff from Schwab, the IBM HVWS team, and the WebSphere Development and Architecture/Performance teams collaborated to design and execute the tests of the Barista architecture. Kruse notes that "the collaboration among teams was exceptional not only because our objectives were met, but also because Schwab and IBM jointly pushed the envelope on what was currently understood as possible, and each came away with knowledge and experience that brings business value to their individual organizations."

Charles Schwab has developed in the Barista architecture, a solid foundation for the future development of their retail electronic brokerage. "The new Barista architecture will improve our ability to scale our Web site economically, as well as provide more consistent services to our customers," says Kruse. The architecture, based upon best of breed technology such as Java, WebSphere and DB2, proved to exceed all expectations for scalability, stability, performance, and manageability.

The environments for different Barista tests were similar. The primary difference was the number of servers required for the types of testing. Barista testing focused on stress tests of the architecture, and so required many servers. For example, in one of the scenarios IBM created an unusually large single WebSphere domain consisting of 44 servers in the presentation tier, 15 servers in the business logic tier, and 15 servers in the back-end. Barista testing required a smaller configuration as the focus was on testing the behavior of a specific application.

Figure 8-10 shows the topology of the large-scale Barista test environment.

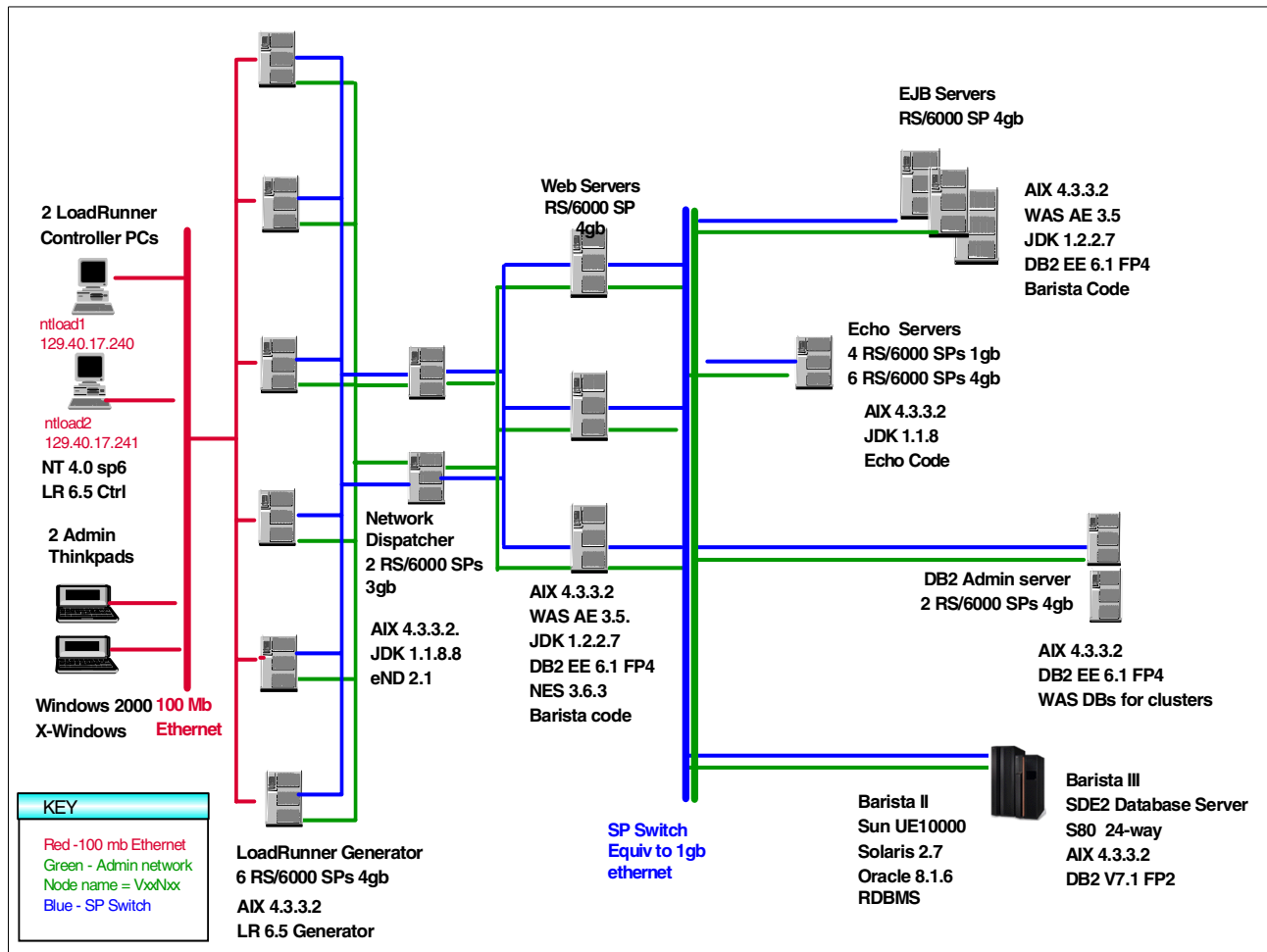


Figure 8-10 Topology of the Barista test environment

Table 8-1 summarizes the hardware used.

Table 8-1 Hardware used


Device	Quantity	Hardware Definition
Web server and database servers	60	IBM RS/6000 Scalable POWERparallel Systems (RS/6000 SP)
Echo servers	15	RS/6000 SP
DB2 database	1	IBM pSeries S80 24-way processor
Oracle database	1	Sun Solaris
9077 router	1	For database connectivity
LoadRunner driver	6	RS/6000 SP
LoadRunner controller	2	PC with Microsoft Windows NT
Administration laptops	2	IBM ThinkPads
Administration server	2	RS/6000 SP
Network Dispatcher	2	RS/6000 SP

Software and test tools

- ▶ WebSphere Application Server 3.5.3 Image GA Version
- ▶ Java Development Kit (JDK) 1.2.2, 1.1.8 plus Service Level 8
- ▶ AIX 4.3.3.2
- ▶ DB2 Universal Database (UDB) 7.1 Fixpack 2 – for database
- ▶ DB2 UDB 6.1 Fixpack 4 – for WAS repositories
- ▶ eND Network Dispatcher 2.1 (now a component of WebSphere Edge Server)
- ▶ Netscape Enterprise Server 4.6.1
- ▶ Mercury Interactive LoadRunner 6.5.2 (controller for Microsoft Windows NT, generator for AIX)
- ▶ Microsoft Windows NT 4.0 with Service Pack 6 – For LoadRunner Controllers

Monitoring tools

- ▶ AIX -- PTX
- ▶ AIX -- topas
- ▶ AIX -- svmon
- ▶ Mercury Interactive LoadRunner Analyzer
- ▶ WebSphere Resource Analyzer
- ▶ WebSphere container tracing
- ▶ WebSphere Thread Analyzer
- ▶ Live Jinsight – Java Profiler



Improving the scalability of a WebSphere application with multihome servlets

Successful e-businesses are those that can adapt quickly and efficiently to the demands of their business growth. Many such businesses found that the single tier Web environment that was a satisfactory configuration for their initial customer volumes can no longer meet their business needs. As e-businesses experience growth in customers and customer transactions, the Web environment infrastructure must evolve, or scale, to satisfy the increasing demands. The requirement for scalability can also be driven by application inefficiencies, a fairly common occurrence in new online applications. In either case, the scalability of an application and its environment becomes a key factor in an e-business' success.

The technical approaches to achieving scalability are typically vertical scaling (within a system) or horizontal scaling (across systems). This chapter describes multihome servlets, a vertical scaling technique that the High Volume Web Sites (HVWS) team has deployed successfully in customer engagements. The chapter also compares multihome servlets to another vertical scaling technique, servlet cloning. Both techniques seek to optimize resource use by segmenting the resources of the physical system. While servlet cloning provides many scaling advantages, it imposes strict technical prerequisites that can hinder extensive use. The multihome servlets technique is an alternative way of gaining the benefits of servlet cloning - primarily, smaller Java virtual machines (JVMs) that reduce the impact of garbage collection - without incurring the drawbacks - primarily, the requirement to use session persistence.

The implementation of multihome servlets segments a physical machine by creating a virtual host and pairing it with its own servlet (JVM), and then creating multiple copies of these pairs in one physical machine. The creation of virtual hosts enables each virtual host-servlet pair to be uniquely addressable by the dispatching hardware/software. Thus, when the dispatching hardware/software receives a request from the browser application, the dispatcher determines what particular virtual host-servlet pair to direct the request to, thereby distributing the application load across the available virtual host-servlet pairs. Features within the

dispatcher are responsible for sending subsequent requests from each browser session to the appropriate virtual host-servlet pair.

Using the technique of multihome servlets WebSphere customers can quickly and easily improve the scalability of their Web environment, whether the need for additional system throughput is a result of application inefficiencies or volumes growth.

Overview of scaling servlets

Servlets are the Java programs that generate dynamic content and interact with Web clients. Servlets can be implemented in different ways depending on the requirements of the operating environment. The limitations of a single servlet may suggest the need for servlet cloning. When servlet cloning does not yield the necessary scaling, the technique of multihome servlets may be appropriate.

Single-tier e-business applications are relatively simple. A typical configuration for a WebSphere application, shown in Figure 9-1, consists of one Web (HTTP) server and one Web application server (WAS) with one Java virtual machine (JVM) containing servlets and one Enterprise JavaBean (EJB) container. The Web server and Web application server reside on the same machine. The performance of this configuration is limited by the power of the machine and by various constraints inherent in the configuration. As the number of end-users increases, the performance may be seriously diminished by the limited capacity of such a configuration.

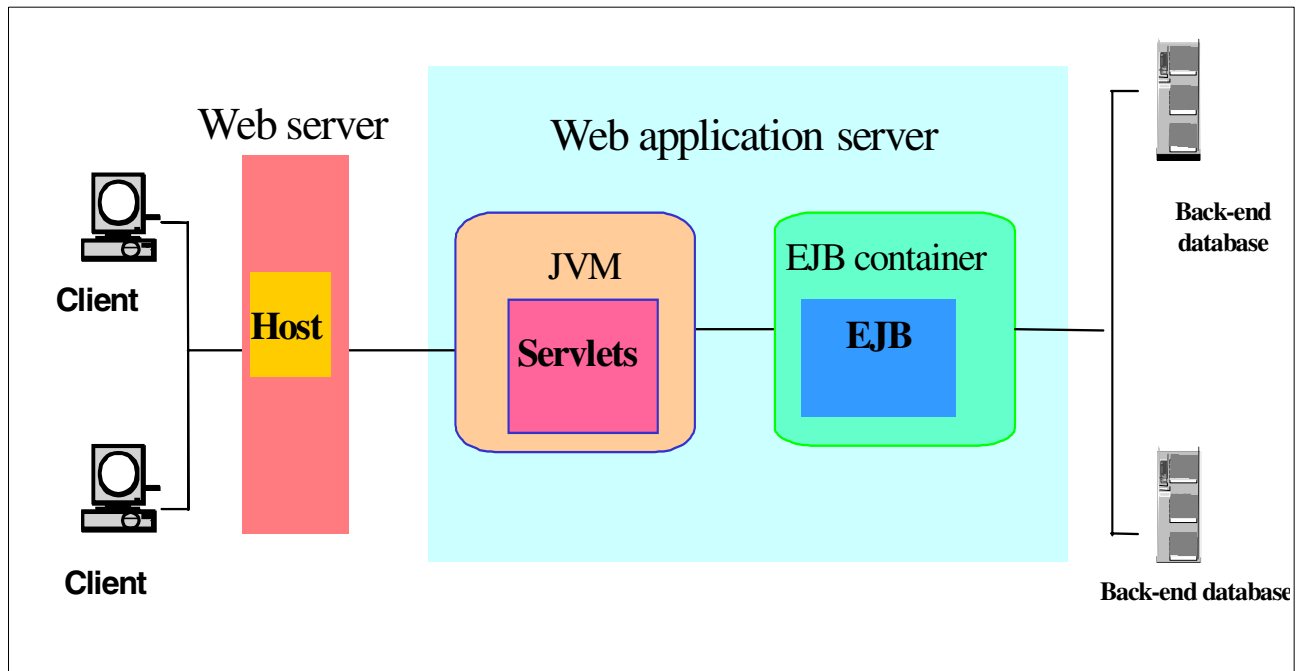


Figure 9-1 Basic WebSphere e-business application configuration

Experience shows that a single application server, implemented by a single JVM process, cannot always fully use the processing power of a large machine. This is particularly true on large multiprocessor machines because of inherent concurrency limitations within a single JVM. The mechanisms available to improve the scalability of a WebSphere application require a configuration implemented with such features as workload management, fail-over, fault isolation, and session management.

The most common and widely used mechanism is vertical cloning, which is the practice of defining clones of an application server on the same physical machine. Vertical cloning provides a straightforward mechanism to create multiple JVM processes that together can fully use the available processing power. Each clone uses its own JVM to provide an identical, but independent, process in which the application runs. Vertical cloning simplifies system administration because clones can be used to quickly create and maintain identical copies of a server configuration. It is a way to organize workload distribution for several mechanisms provided with WebSphere, such as open servlet engine (OSE) remote, workload manager, and servlet redirector.

The implementation of vertical cloning in the JVM/servlet layer is called servlet cloning. Servlet cloning is an easy way to duplicate resources to facilitate workload management and fail-over. Servlet cloning involves using WebSphere administration commands to create a model of an existing JVM, then to create clones of that model. Figure 9-2 shows a single-tier configuration with servlet cloning.

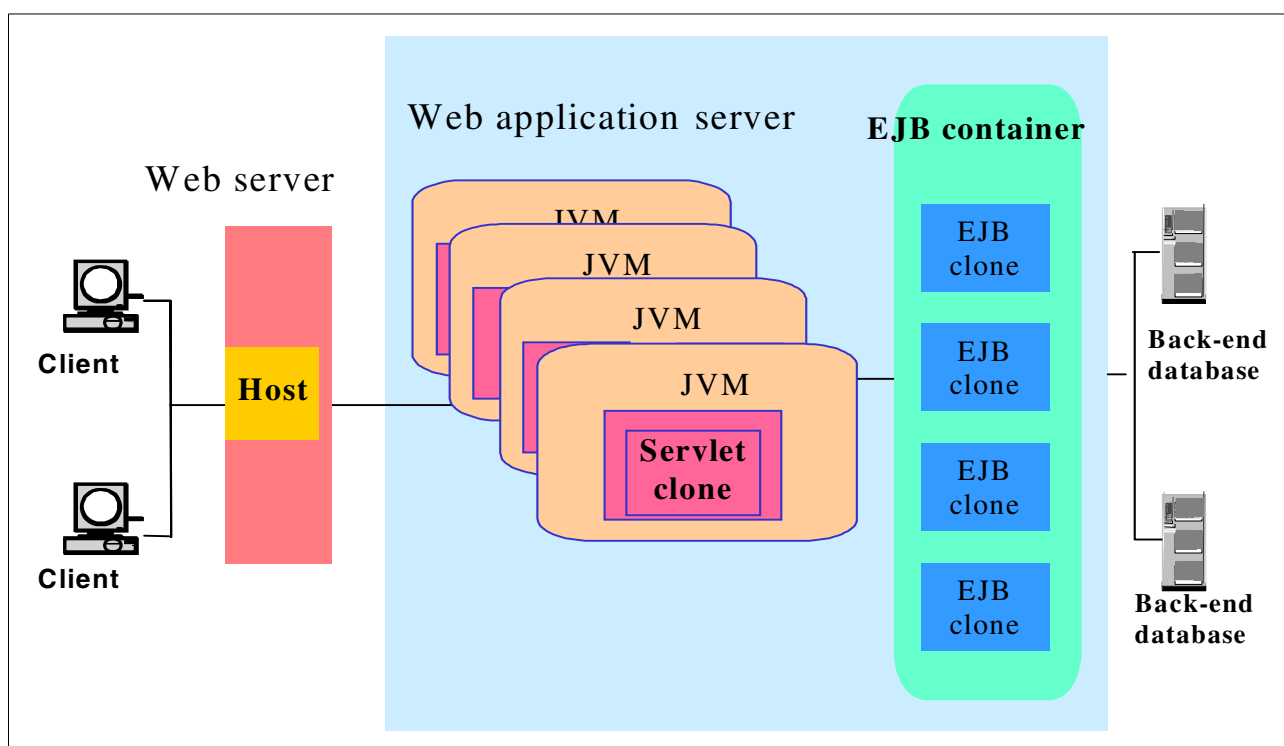


Figure 9-2 Configuration with servlet cloning

The JVM manages the session between the HTTP client and the servlet. The session manager stores session-related information either in memory within the application server --- in which case it cannot be shared with other application servers --- or in a back-end database, shared by all application servers. The latter option is referred to as session persistence.

Session persistence is critical to HTTP sessions with any load distribution configuration. Therefore, servlet cloning requires session persistence. When a back-end database query is involved, performance is compromised because of the overhead needed to facilitate the query. To improve performance, the session manager may implement sophisticated caching optimizations to minimize the overhead of accessing the back-end database. However, to facilitate caching, the overhead also requires WebSphere to use DB2 to cache session data.

Some applications implement their proprietary sessions and don't use the WAS session. Those applications might not work correctly when implementing servlet cloning. In our

customer engagement experience, we've observed that when the customer application is not fit for servlet cloning, symptoms of external problems emerge, for example:

- ▶ Intermittent servlet/JSP generic page errors
- ▶ Failure of application to achieve improvements in throughput (transactions per second or HTTP hits per second)

Therefore, the prerequisite of session persistence imposes challenges when servlet cloning is either not appropriate or does not achieve desired levels of scalability. Such prerequisites limit extensive use of servlet cloning and diminish its acknowledged advantages.

Multihome servlets

Using multihome servlets can reduce the limitations of servlet cloning. Figure 9-3 shows the implementation of multihome servlets. A traditional, single-tier e-business configuration consists of a single physical machine with one IP address serving as host to one JVM/servlet. Servlet cloning enables a single physical machine to serve as host to multiple JVM/servlets. Now a single machine can host multiple IP addresses through the use of virtual hosts, each served by a JVM/servlet.

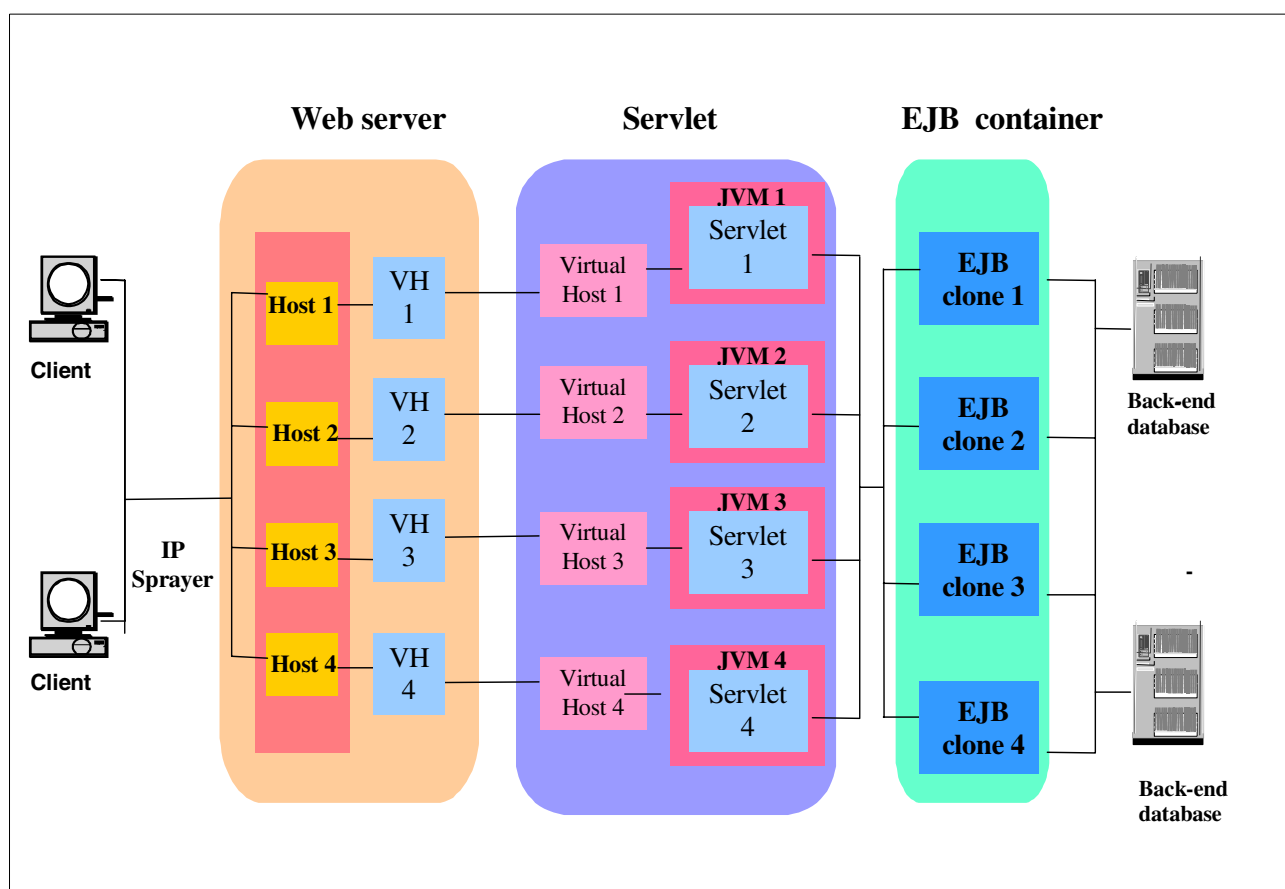


Figure 9-3 Multihome servlet technique

Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine. Even though the virtual hosts share a machine, each one has a unique Web address and runs its own JVM, isolated from other virtual hosts and therefore not requiring workload manager.

Each virtual host has a logical name, for example VH1 and VH2, and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP hostname and port number used to request the servlet. When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet.

The resources of an original servlet application can be associated with WebSphere's VirtualHost1 and the resources of the second copied servlet with WebSphere's VirtualHost2. VirtualHost1 is defined in WebSphere mapping to WebServer VH1, and VirtualHost2 is mapped to WebServer VH2. In our case of using Apache WebServer, both VH1 and VH2 are defined in the server configuration of the same physical machine. Now the two virtual host sites offer the same servlet.

Each host with a distinctive IP address has its own instances of the servlet, which are unaware of instances of the other hosts. An initial browser request is made using a generic VirtualHost Web address, which is routed to a specific VirtualHost1(2) site either by using a hardware local director or a software tool such as WebSphere's Edge Server (Network Dispatcher). Subsequent requests during the same session go directly to the same VirtualHost as a result of enabling Network Dispatcher's sticky feature or implementing the application accordingly.

Benefits of multihome servlets

Multihome servlets can offer significant benefits. They provide the ability to:

- ▶ Circumvent bottlenecks in application memory
- ▶ Improve scalability
- ▶ Implement a tactical solution and maintain important business opportunities

Bottlenecks in application memory are common in online applications developed in-house. Multihome servlets are a way to avoid bottlenecks while achieving scalability and maintaining high performance. Many HVWS customer engagements involve integrating in-house applications with the WebSphere application server. Because the e-business requirements for the application are new and application developers are still learning how to address them, memory bottlenecks are common. A major cause of memory bottlenecks is what is called memory leak, a condition that cannot always be fixed before putting an application into production.

When there is memory leak, the application requires more memory and a correspondingly larger JVM heap. The time it takes for Java to compact or reclaim space (garbage collection) is a product of heap size. As heap size increases, application response time becomes long and unstable. Because the multihome servlets segment the original heap size, the heap for each servlet is smaller. When the size of a heap is reduced, there is a corresponding reduction in the time for garbage collection and performance is maintained.

An application bottleneck can also result from the low-level design of the application, which involves frequency of certain function calls, interface to the network layers, basic data type choices, and thread processing procedures. Removing the bottleneck could involve redesigning the architecture. Many customers can't afford to mount that level of effort and indefinitely delay deploying the application. While such a delay probably results in a better technical application, it can expose the company to the considerable risk of losing their online business opportunities to their competition.

The next section describes the experience of the HVWS team with a major bank. Managing a bottleneck as described above enabled the bank's application to scale and function as

required in the production environment. With the application in production, the customer is working toward its business objectives, while the development team diagnoses and corrects the root cause of the bottleneck. Multihome servlets provide a good interim solution, adding some tasks to application deployment and maintenance.

Case study

In a recent customer engagement, the HVWS team faced a bottleneck in application memory. The project was to assess the performance and scalability characteristics of an online trading system for a major commercial bank. The CPU utilization on the Web or Web application (EJB) servers was stuck at 40-50 percent. If the load increased slightly, the response time rose by a factor of five; sometimes the application terminated. The logins per second, which was the more important performance criterion, never exceeded seven. However, the bank's objective for the test environment was twenty logins per second. Vertical cloning diminished performance even more because of the overhead required given the application did not support session persistence.

Performance improved significantly with multihome servlets. Multihome servlets avoided the application bottleneck and met the business objectives set by the bank's executives. Two configurations were tested: the first configuration (3:2) had a ratio of three servlet machines called Web servers to two EJB machines called Web applications servers; the second configuration (3:3) had a ratio of three servlet machines to three EJB machines. With multihome servlets, the CPU utilization increased to over 90%, the logins per second rose by 144%, and the total testing load doubled. Here is the summary of the test results.

Multihome servlets improved the scalability of the application while maintaining satisfactory performance. The following charts compare the performance of applications with a single servlet and four multihome servlets in a 3:2 configuration.

Figure 9-4 compares the key performance data of the customer application. In this test, we tried to drive the single servlet application to the maximum of 1125 users. Response time degraded to almost ten seconds and CPU utilization stuck at 70%. The number of quotes per second, one of the key business criteria, was far below the customer requirement. With four multihome servlets, we can drive 1140 users with 4.9 seconds in response time and 80 to 90% CPU utilization. The number of quotes per second was six times higher.

Criteria	Test Results with 1 Servlet	Test Results with 4 Servlets
Total Users	1125	1140
JVM Used	1	4
Login Rate	11.4 /sec	16.2 /sec
Quote Rate	68 /sec	424 /sec
Servlet CPU	60%	80.4%
EJB CPU	70%	89%
Login response time	9.8	4.9 sec
Quote response time	5.5	2.5 sec

Figure 9-4 Performance comparison of single servlet application versus multihome servlets application

The measures associated with login and quote transactions were of primary interest to the customer because login consumes the most resources and quote is the most important function of the application. Figures 9-5 and 9-6 compare the performance of the key functions

of login and quote. The figures show that the multihome servlet application improved significantly: transactions per second increased and response time decreased.

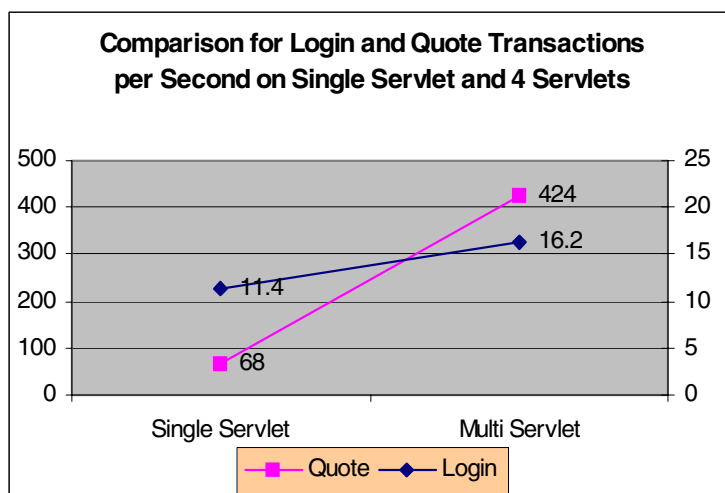


Figure 9-5 Performance comparison of single servlet application versus multihome servlets application on transactions per second of key business criteria

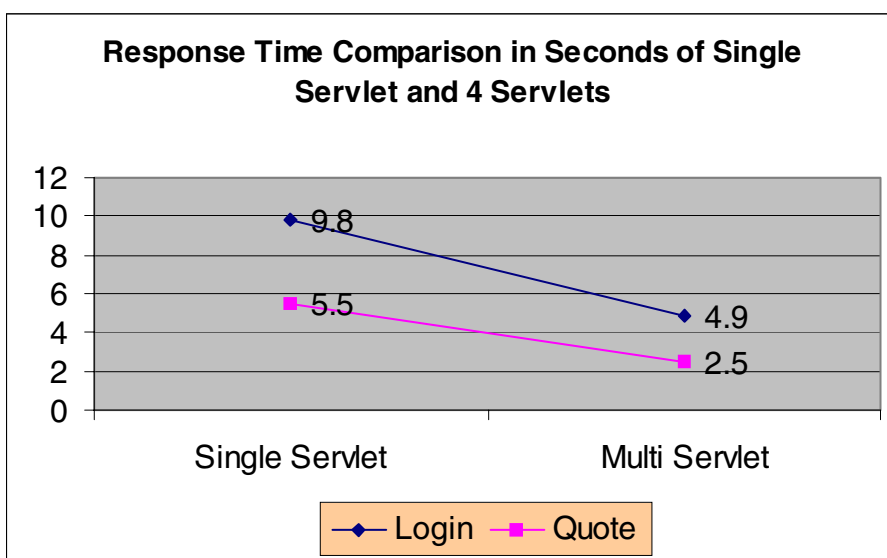


Figure 9-6 Performance comparison of single servlet application versus multihome servlets application on response time of key business criteria

Setting up multihome servlets

This section summarizes how we set up the multihome servlets. The information is for your reference only and the details might be different in each configuration.

Prerequisites

- Install WebSphere 3.5 or above in end-to-end, single host configuration.
- Confirm the application executed without errors.
- Shut down the Web servers, the Web application servers, and the WebSphere domain.

Setup

Figure 9-7 summarizes the steps required to set up multihome servlets.

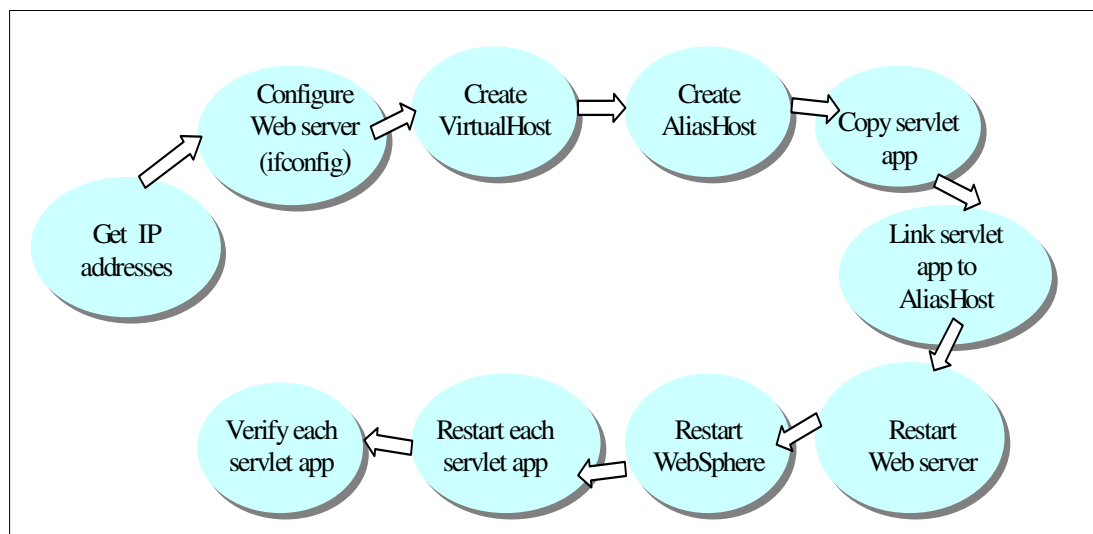


Figure 9-7 Setting up multihome servlets

1. Obtain an additional IP addresses for each virtual home. This allows a Web server to respond to requests for more than one server address.
2. Configure the Web server to accept IP packets for multiple addresses. This can be accomplished with the ifconfig alias flag.
3. Create new VirtualHost for each additional virtual home in WebServer configuration. This can be done in the HTTP configuration file, httpd.conf, of the IBM HTTP server or Apache Web server, as follows:

```
<VirtualHost hostxx.some_domain.com>
    DocumentRoot /www/docs/hostxx.some_domain.com
    ServerName hostxx.some_domain.com
    ErrorLog logs/hostxx.some_domain.com-error.log
    TransferLog logs/hostxx.some_domain.com-access.log
</VirtualHost>
```

4. For a Microsoft IIS server, in IIS admin console, go to Action. Click Advanced on the Web Site tab. Enter each IP address and its corresponding port number.
5. Use either XML or GUI Admin Console to create new AliasHost for each virtual home in the WebSphere configuration.
6. Use the WAS batching facility, XML export/import facility, or online GUI Admin console to create the multihome servlet by copying the original servlet application to as many new servlet application names as needed.
7. Associate each new copied servlet application with a unique AliasHost created above.
8. Restart the Web server, which will incorporate the new IP addresses and associated changes.
9. Restart the WebSphere node. The admin console will reflect the changes and new servlet applications.
10. Start each servlet application through the admin console.
11. Verify that each servlet application works and responds to browser requests. We may try each IP address from the browser and see if all the IP addresses work correctly.

Summary

The HVWS team developed the multihome servlet technique when confronted with the limitations imposed by the prerequisite of servlet cloning for session persistence. Multihome servlets offer a useful alternative to servlet cloning. Their benefits include the ability to:

- ▶ Circumvent bottlenecks in application memory
- ▶ Improve scalability
- ▶ Implement a tactical solution and maintain important business opportunities

Multihome servlets offer these benefits when servlet cloning is either not appropriate or does not achieve desired levels of scalability. This chapter reviewed the circumstances that led to the development of the multihome servlet technique. It describes their benefits and a procedure for setup.

References

WebSphere Scalability: WLM and Clustering, using WebSphere Application Advanced Edition, SG24-6153-00, at:

<http://www.ibm.com/redbooks>

Design for Scalability, An Update, September 2001, at the IBM High-Volume Web Site:

<http://www.ibm.com/websphere/developer/zones/hvws>

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ WebSphere Application Server Performance Web site provides access to helpful performance reports, tools, and downloads
<http://www.ibm.com/software/webservers/appserv/performance.html>
- ▶ High Volume Web Sites
<http://www.ibm.com/websphere/developer/zones/hvws>
- ▶ Failover and Recovery in WebSphere Application Server Advanced Edition 4.0
<http://www7.software.ibm.com/vadd-bin/ftpd1?1/vadc/wsdd/pdf/modjeski.pdf>
- ▶ IBM Tivoli performance and availability tools
<http://www.tivoli.com/products/solutions/availability/news.html>
- ▶ IBM WebSphere software platform for e-business includes edge servers, Web application servers, development and deployment tools, and Web applications
<http://www.ibm.com/websphere/developer>
- ▶ IBM WebSphere Commerce Suite, used by customers who run large-scale online shopping sites
<http://www.ibm.com/software/webservers/commerce>
- ▶ Software used by trading sites studied using WebSphere Application Server
<http://www.ibm.com/software/webservers/appserver>
- ▶ Software used by trading sites studied using, WebSphere MQSeries
<http://www.ibm.com/software/software/ts/mqseries>
- ▶ Download a demo version of PageDetailer, the tool in WebSphere Studio that measures in detail every element in a page download to assists in performance analysis and optimization
<http://www.ibm.com/software/webservers/studio/download.html>
- ▶ Carnegie Mellon Software Engineering Institute
<http://www.sei.cmu.edu>

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the **CD-ROMs** button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

A

- additional scaling techniques 19
- AIX performance tools 94
- application response times - baseline versus peak 83
- availability ix, 60
 - assure your IT processes promote availability 66
 - common
 - inhibitors 64
 - techniques 65
 - components for consideration 70
 - concepts and costs 60
 - continuous availability 60–61
 - create hardware and software clusters/implement load balancing 69
 - design your applications to enhance availability 65
 - design your infrastructure for availability 68
 - e-business infrastructure with availability measures 62
 - high availability 60
 - implementing connectors 72
 - introduction 60
 - investments in availability vary by workload pattern 64
 - making trade-offs 63
 - minimum topology for a highly available WebSphere configuration 71
 - multiple physical sites to protect against the effect of disasters 73
 - multi-tier infrastructure designed for high availability 69
 - protect your data from accidental loss, viruses, and disasters 72
 - SANs change how data is accessed and increases availability 73
 - trade-offs between costs and consequences of an outage 63
 - unavailability is expensive 62
 - versus actual time lost 61
- availability versus actual time lost 61
- average cost per Web transaction 88

B

- basic WebSphere e-business application configuration 120
- benefits of multihome servlets 123
- best practices viii

C

- capacity planning 40
 - analyze trends and set performance objectives 41, 49
 - identify your workload pattern 41
 - measure performance of current site 41–42

- model your infrastructure alternatives 41, 51
- character of workload
 - amount of cross session information 8
 - data volatility 9
 - high volume dynamic transactional fast growth 8
 - number of page views 9
 - number of unique items 9
 - other workload character items 9
 - percent secure pages (privacy) 9
 - transaction complexity 8
 - transaction volume swing 9
 - use of security (authenticate, integrity, nonrepudiate) 9
 - volume of dynamic searches 8
 - volume of user specific responses 8
- characterizing your workload 8
- Charles Schwab
 - Barista 98
 - EJB test scenario 107
 - introducing the project 106
 - test environment
 - hardware 118
 - monitoring tools 118
 - software and test tools 118
- benchmark
 - environment 102
 - software configuration 103
- best practices 114
- e-business today 97
- fine-tuning the scalability of a multi-tier architecture 105
- IBM benchmark 99
- making sure the new architecture measures up 98
- page views per second 100
- proposed architecture Barista 98
- results
 - Barista EJB multidomain test scenario 108
 - Barista EJB single domain test scenario 109
 - Barista soak test 113
 - Barista test (one presentation server and one business logic server) 111
 - Barista test (two presentation servers and one business logic server) 112
 - summary of Barista results 113
- Schwab response to the benchmark results 100
- technical benchmark details 101
- test scenarios 102
- testing Barista 107
- topology
 - Barista EJB test 108
 - Barista test (one presentation and one business logic) 110
 - Barista test (two presentations and one business logic server) 111
 - Barista test environment 107, 117

- Charles Schwab puts growth plan to the test 97
- choosing between two and three tiers 22
- common pitfalls
 - architecture phase 21
 - deployment phase 22
 - design phase 21
 - development and test phases 22
 - planning phase 21
 - summary 21
- comparing multihome servlets to servlet cloning 119
- components most affected 9
- configuration with servlet cloning 121
- continuous availability 61
- customer
 - engagements viii, xi
 - self-service sites let users help themselves 6

D

- design 27
 - pages for performance 27
 - practices that can improve performance 34
 - scalability 11

E

- each workload pattern has an associated class of user requests 44
- e-business infrastructure with availability measures 62
- emerging standards and technologies 23
- example
 - seasonality demonstrated by one week from Nagano 46
 - transition matrix for an online shopping visit 49
 - two-, three-, and four-tiered infrastructures 23
 - Web page metrics 43

F

- fine-tuning the scalability of a multi-tier architecture 105

H

- high availability x
- high performance ix
- High Volume Web Sites (HVWS)
 - locations 138
- High-Volume Web Sites (HVWS) vii, 138
 - locations vii
- hit rates over six months for a financial site 87
- how latency varies based on workload pattern and tier 15
- how scaling techniques relate to scaling objectives 13

I

- IBM Tivoli Monitoring for Transaction Performance 93
- IBM Tivoli Monitoring for Web Infrastructure 91
- IBM Tivoli Performance Viewer 91
- IBM Tivoli Web Site Analyzer 92
- IBM Wimbledon Web site on its record-breaking day 45
- improving the scalability of a WebSphere application with multihome servlets 119

- infrastructure component
 - data servers 9
 - edge server 9
 - network 9
 - security servers 9
 - transaction servers 9
 - web application server 9
- introducing
 - methodology for capacity planning 40
 - scalability 12
 - Web communications 28
 - workload patterns 6
- investments in availability vary by workload pattern 64
- It is not just about satisfying customers 37

J

- J2EE 23

K

- know your workload 3

L

- life cycle of a Web site viii
- Linux 23

M

- managing Web site performance 79
- maximize Web site availability 59
- measure performance of current site 42
- methodology
 - capacity planning 40
 - managing performance xi
 - managing performance of a HVWS 81
 - modeling x
- metrics for page hits per day 5
- minimum topology for a highly available WebSphere configuration 71
- models for HVWS capacity planning 52
- multihome
 - servlet technique 122
 - servlets 119, 122
- multihome servlets case study 124
- multi-tier
 - infrastructure designed for high availability 69
 - infrastructure for e-business vii, 40

O

- online shopping
 - sites let users browse and buy 6
- online shopping script 47
- online shopping site measurements 47
- optimizing for scalability ix
- overview
 - scaling servlets 120
 - Web communications 29

P

- page design practices
 - manage
 - load sequences 37
 - number of connections 35
 - number of servers accessed 36
 - number, size, and complexity of items 34
 - use of white space 36
 - understand impact of data security 37
- page download measurements 30
- page load time rankings 31
- pattern
 - categories/examples 7
 - content 7
 - cross-session info 8
 - data volatility 8
 - legacy integration/complexity 8
 - page views 8
 - percent secure pages 7
 - searches 8
 - security 7
 - unique items 8
 - volume of transactions 8
- performance comparison
 - single servlet application versus multihome servlets application 124
 - single servlet application versus multihome servlets application on response time of key business criteria 125
 - single servlet application versus multihome servlets application on transactions per second of key business criteria 125
- performance management
 - AIX performance tools 94
 - analyze and tune components 83
 - application response times - baseline vs peak 83
 - average cost per Web transaction 88
 - comparison
 - single servlet application versus multihome servlets application 124
 - cost savings with proposed enhancement 89
 - current and projected system load 89
 - establish performance objectives 81
 - hit rates over six months for a financial site 87
 - IBM Tivoli Monitoring for Transaction Performance 93
 - IBM Tivoli Performance Viewer 91
 - IBM Tivoli Web Site Analyzer 92
 - monitor and measure the site 82
 - performance comparison of single servlet application versus multihome servlets application on response time of key business criteria 125
 - performance comparison of single servlet application versus multihome servlets application on transactions per second of key business criteria 125
 - predict and plan for the future 85
 - retail customer seasonal peaks 86
 - retailer usage patterns over one week 86
 - scenarios 87
 - tools available to monitor and analyze Web site com-

- ponents 85
- performance monitoring
 - IBM Tivoli Monitoring for Web Infrastructure 91
- pervasive computing devices 24
- plan for growth 39
- projected measurements for online shopping site 49
- publish/subscribe Web sites provide users with information 6

R

- Redbooks Web site 130
 - Contact us xii
- reliability ix
- requests per second over one hour during Nagano Olympics 50
- retail
 - customer seasonal peaks 86
 - site with seasonal peaks 4
 - usage patterns over one week 86

S

- sample graph showing components of performance 55
- sample measurement of a Web page 32
- scalability/performance curves 13
- scaling a WebSphere Commerce Web site 54
- scaling a WebSphere online trading site 19
- scaling techniques
 - aggregate user data 17
 - batch requests 16
 - cache 17
 - create a cluster of machines 16
 - manage connections 17
 - segment the workload 16
 - use a faster machine 16
 - use appliance servers 16
- scaling techniques applied to components 18
- security ix
- self-managing servers 24
- servlets
 - basic WebSphere e-business application configuration 120
 - benefits of multihome servlets 123
 - configuration with servlet cloning 121
 - multihome 119, 122
 - case study 124
 - technique 122
 - overview of scaling servlets 120
 - performance comparison
 - single servlet application versus multihome servlets application on response time of key business criteria 125
 - single servlet application versus multihome servlets application on transactions per second of key business criteria 125
 - setting up multihome servlets 125
- setting up multihome servlets 125
- site performance
 - obtain site measurements 47
 - seasonality 46

- understand workload metrics 43
- site type
 - customer self-service 7
 - online shopping 7
 - publish/subscribe 7
 - trading 7
 - Web services/B2B 7
- steps to scaling infrastructure
 - apply the techniques 18
 - categorize your workload 15
 - determine the components most affected 15
 - reevaluate 19
 - select the scaling techniques to apply to scale the workload 16
 - understand the application environment 14
- storage area networks (SAN) change how data is accessed and increases availability 73
- summary of the eight scaling techniques 16

T

- tools available to monitor and analyze Web site components 85
- tools for monitoring performance 91
- trade-offs between costs and consequences of an outage 63
- trading sites let users buy and sell 6
- traffic patterns from Nagano Olympic Games 45
- traffic scaling for peak hours in different years 51
- typical Web site loads over a 24-hour period 3

U

- unavailability is expensive 62
- understanding your workload 7
- using Web services B2B sites buy from/sell to each other 7

W

- Web services 23
- Web site availability 59
- Web site classifications 7
- Web site with multiple tiers 53
- WebSphere Edge Server for Multiplatforms 71
- what is a good page 33
- when bad things happen to good pages 30
- workload patterns 6
 - customer self-service sites let users help themselves 6
 - online shopping sites let users browse and buy 6
 - publish/subscribe Web sites provide users with information 6
 - trading sites let users buy and sell 6
 - using Web services B2B sites buy from/sell to each other 7
- workload patterns and Web site classifications 7



Best Practices for High-Volume Web Sites

**Designing for
scalability**

Planning for growth

**Managing Web sites
for performance**

For more than three years, IBM's High-Volume Web Sites (HVWS) team has been working with many of the world's largest Web sites. The team has accumulated a significant amount of knowledge and defined best practices for designing and deploying high-volume sites, earning a reputation as one of the world's leading centers of expertise on scalable e-business infrastructures. The team has locations in California, New York, Japan, Korea, China, Taiwan, and the United Kingdom.

The IT infrastructures that comprise most high-volume sites present unique challenges in design, implementation, and management. While actual implementations vary, a typical e-business infrastructure is comprised of several tiers. Each tier handles a particular set of functions, such as serving content (Web servers, such as the IBM HTTP Server), providing integration business logic (Web application servers, such as the WebSphere Application Server), or processing database transactions (transaction and database servers). Site workloads are assumed to be high volume, serving dynamic, volatile data. As it accumulates experience and knowledge, the HVWS team publishes papers aimed at helping CIOs and others like you understand and meet the new challenges presented during one or more of the phases. This redbook is a compilation of the HVWS papers, which are available individually at the HVWS Web page:

<http://www.ibm.com/websphere/developer/zones/hvws>

where you will find the latest on the HVWS teams consultative project documentation.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6562-00

ISBN 0738425095