IBM

Tuning Linux OS on System p

The POWER of Innovation

- Linux kernel tuning
- DB2, WebSphere, and Oracle tuning
- SystemTap examples

Stephen Hochstetler
Kraml Liu
Andre Metelo
Prasanna S Panchamukhi
John Zucker

Redbooks

ibm.com/redbooks

International Technical Support Organization

Tuning Linux OS on IBM System p
The POWER of Innovation

June 2007

SG24-7338-00

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

First Edition (June 2007)

This edition applies to Red Hat Enterprise Linux AS 4 and SUSE Linux Enterprise Server 10.

Contents

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 illustrate 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.

Trademarks

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

AIX 5L™
AIX®
BladeCenter®
Chipkill™
DB2®
developerWorks®
eServer™
GPFS™
HACMP™
i5/OS®
Informix®
IntelliStation®
IBM®
Lotus®
Micro-Partitioning™
OpenPower™
PowerPC®
POWER™
POWER Hypervisor™
POWER4™
POWER5™
POWER5+™
pSeries®
Redbooks®
Redbooks (logo) ®
Service Director™
System i™
System p™
System p5™
System x™
System z™
Tivoli®
TotalStorage®
WebSphere®

The following terms are trademarks of other companies:

SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

EJB, Java, Java Naming and Directory Interface, JDBC, JDK, JRE, JSP, JVM, J2EE, J2SE, Power Management, Solstice, SOAP with Attachments API for Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Intel, Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

This IBM® Redbooks® publication will help you install, tailor, and configure solutions based on the Linux® OS on the IBM System p™ hardware. The intended audience is the person that is installing the software and configuring both the applications and the operating system. You can do both Linux kernel tuning and application configuration. These configuration options can be done whether you are using the advanced virtualization features of System p servers or not.

When you use the advanced virtualization features, additional considerations should be reviewed to enhance the performance of the system. With virtualization and a server consolidation configuration, you can share resources such as CPU across multiple partitions so that the resources are supplied to the workload that currently needs them. The configuration of the Virtual I/O Server is important for this setup and the Linux installations on the partitions to work properly.

This book also looks at specific solutions and applications (WebSphere®, DB2®, and Oracle®) and makes recommendations for configuration. You will see what we configuration we used so that you can correct your own bottlenecks and determine which configuration items to tune. This is the most valuable part of the book, since each systems's configuration and workload will be unique.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Stephen Hochstetler is an IT Specialist at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on all areas of Linux, System p servers, and Tivoli®. Before joining the ITSO six years ago, Stephen worked in Tivoli Services, USA as a network management specialist.

Kraml Liu is a Linux on POWER™ client support engineer of IBM Technology Product (Shenzhen) Company Ltd. in China. He has seven years of experience in the Linux environment and two years of experience with System p servers. He currently provides pre-sales technical support for clients running Linux on IBM System p, OpenPower™, and Blade J20/JS21, and training for clients and IBM Business Partners. His areas of expertise include Linux problem determination and tuning, open source and commercial applications for Linux, and System p hardware.

Andre Metelo is a Senior, Certified IT Specialist in the Latin America Techline team in the IBM S&D Technical support division in the United States. He has seven years of experience in the System p field as well as three and a half years in the networking field. He has worked at IBM for 10 years. His areas of expertise include System p hardware, Linux, AIX®, TCP/IP networking, and ATM. He holds a degree in Industrial Engineering from the Univervisade Federal do Rio de Janeiro. He has published articles in the *CPU* magazine in Brazil, as well as several IBM internal and externals Techdocs.

Prasanna S Panchamukhi is a Senior Staff Software Engineer in System Technology Group, IBM India Private Ltd. He has about four years of experience in Linux kernel Reliability, Availability, and Serviceability tools. He holds a degree in Mechanical Engineering from Karnataka University Dharwar. His areas of expertise include Linux trace/probe tools, kernel debugger tools, and crash dump. He has published several papers at Ottawa Linux, Linux Kongress, and Linux Banagalore conferences on kprobes and user-space probes, and also published articles on IBM developerWorks® on kprobes and oprofile.

John Zucker is a Senior IT Specialist working in the IBM Innovation Centre in the UK. He has been working on application software for IBM servers since joining IBM 11 years ago. He holds a doctorate in Knowledge Base Programming applied to engineering design and a Master's degree in Computing Science. He has experience revolving around J2EE™ application development and scalability mainly with reference to UNIX® systems. He currently provides developers from IBM Business Partners with architectural and technology briefing that lead to testing and validation of their software. In particular, he enables developers to migrate application software onto Linux on System p.

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

Dr. Balaji Atyam, Cesar Diniz Maciel, Dr. Carlos Sosa, Nick Harris, Randy Kuseske, Dan Lacine, Ralph Cooley, Bob Walte, David Watts, Greg McKinght, Martha Centeno
IBM USA

Arsi Kortesniemi
IBM Finland

Mattew Jenner
IBM Australia

Frank Berres
IBM Germany

Bruno Blanchard, Nicholas Guerin, Francois Armingaud, Jose Rodriguez Ruibal
IBM France

Pedro Coelho
IBM Portugal

Lei Lui
IBM China

Ravikiran Thirumalai
IBM India

Massimiliano Belardi
IBM Italy

Tomoni Takada
IBM Japan

Alexander Zaretsky
IBM Colombia

Eduardo Ciliendo
IBM Switezerland

Brian Jeffery
IBM UK

Nic Irving
Computer Science Corporation, Australia

Ben Gibbs
Technonics Inc, USA

Lancelot Castillo
Questronix Corporation, Philippines

Carmeron Hunt
IT Consultant, USA

Mauricio Lima, Luciano Magalhaes Tome
EDS, Brazil

William Malchisky Jr.
Effective Software, USA

Jim Taylor
IT Consultant, Canada

Dirk Webbeler
DW Solutions, Germany

Marcelo Baptista
Banco Itau, Brazil

Jean-Jacques Clar
Novell, USA

Philip Dundas
Coca-Cola, Australia

Frank Pahor,
Suncorp, Australia

Raymond Philips
Microsoft®, Canada

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbooks publication dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and client satisfaction. As a bonus, you will 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 book form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Introduction to the book

The server is the heart of the entire network operation. The performance of the server is a critical factor in the efficiency of the overall network and affects all users. While simply replacing the entire server with a newer and faster one might be an alternative, it is often more appropriate to replace or to add only to those components that need it and to leave the other components alone.

Often, poor performance is due to bottlenecks in individual hardware subsystems, an incorrectly configured operating system, or a poorly tuned application. The proper tools can help you to diagnose these bottlenecks and removing them can help improve performance.

For example, adding more memory or using the correct device driver can improve performance significantly. Sometimes, however, the hardware or software might not be the direct cause of the poor performance. Instead, the cause might be the way in which the server is configured. In this case, reconfiguring the server to suit the current needs might also lead to a considerable performance improvement.

1.1 Operating an efficient server: four phase

To operate an efficient server, you need to follow these four phases:

1. Have an overall understanding of the environment.

   There are many components within the network environment that can impact server performance and that can present themselves as potential bottlenecks. It is important to understand the role that the server has to play in this environment and to understand where it is located in the network and in relation to other servers in the environment.

2. Pick the correct server for the job.

   After you have established a need for a new server, it is important to have components that allow sufficient bandwidth through those critical subsystems. For example, a file server needs a disk subsystem and a network subsystem that provide sufficient bandwidth for client needs.

3. Configure the hardware appropriately and eliminate initial bottlenecks.

   After you have selected the server hardware (and application software), you need to configure the subsystems (for example, the stripe size on the RAID array and RAID levels) to maximize performance. To ensure that you are actually improving performance, you need to take initial performance readings (called baseline readings) and then compare those with readings taken after you have implemented your changes.

4. Capture and analyze on-going performance data to ensure that bottlenecks do not occur.

   When the server is in production, you need to continue to gather and process performance figures to ensure that your server is still at a near-optimal configuration. You might need to add specific hardware upgrades, such as memory, to achieve this optimal configuration.

   As well as looking at the current situation, it is also appropriate that you perform trend analysis so that you can recognize future bottlenecks before they occur. Trend analysis allows you to plan for hardware upgrades before they are actually needed.

Performance monitoring and tuning is an on-going task. It is not reasonable to simply tune a server once and then assume that it will remain tuned forever. Because the server workload mix changes, so do the location and appearance (and disappearance) of bottlenecks.

1.2 Performance tuning guidelines

Table 1-1 lists guidelines to assist with server management and performance tuning. Although not directly applicable to tuning, following these guidelines should assist in preventing bottlenecks and identifying bottlenecks.

Table 1-1 Performance tuning guidelines

Guidelines	Reasoning
Centralize servers where possible.	Assists with management and can isolate components such as WAN.
Minimize the number of server types.	Enables you to focus on specific subsystems within specific server types.
Standardize configurations.	Enables you to focus on specific subsystems within specific server types.
Use industry accepted protocols and standards.	Prevents attempts to identify bottlenecks with obscure third-party products and tools.
Use appropriate tools.	Fit-for-purpose tools assists with subsystem monitoring and bottleneck analysis.

1.3 Using information from the System p Performance Lab

IBM puts significant effort into ensuring that its servers have the highest performance level possible. Part of this effort is the *System p Performance Lab*, a group in Austin, Texas, where work is done on System p servers through the development phase and after the servers become publicly available.

During the development phase, the lab creates performance models using subsystem and system functional specifications, chip functional specifications, input from the IBM development engineering departments, as well as trace information from the performance lab to do the following:

- Optimize the performance of the subsystem and system before the product is manufactured.
- Make design decision trade-offs.
- Select the optimum performance among various available chipsets that are intended to be used as part of the subsystem or system.
- Select the optimum settings of the chipset parameters.

This information is used to provide subsystem and system design guidance to the development engineering departments.

As the system development phase nears completion, performance measurements are made with prototype subsystems and systems as well as with ship-level systems to do the following:

- Perform stress testing.
- Validate product functional specifications.
- Validate the subsystem and system performance models.
- Optimize the performance of the subsystem and system.
- Improve the performance of third-party vendors tools, adapters and software packages to perform well on System p servers.
- Develop performance white papers for marketing, demonstrating the competitiveness of the System p servers.
- Develop performance tuning guides for clients using specified applications.

Marketing and sales departments and vendors use this information to sell the System p servers and clients can use this information to select the appropriate system and to tune their systems for their applications.

To provide performance data, the System p Performance Lab uses the following benchmarks:

- SAP® Standard Application SD Benchmark 2-Tier / 3-Tier
- TPC-App Benchmark Application Server and Web Services
- TPC-C Benchmark (TPC-C) Transaction processing
- TPC-E Benchmark (TPC-E) Transaction processing
- TPC-H Benchmark (TPC-H) ad hoc decision support
- TPC-DS Benchmark (TPC-DS) decision support
- Oracle 10g RAC Analysis

- SPECweb2005 (World Wide Web server Content)
- SPEC CPU2000 Floating point and integer benchmarks
- SPECjbb2005 (Java™ Business Benchmark)

1.4 Linux Technology Center

The Linux Technology Center (LTC) serves as a center of technical competency for Linux both within IBM and externally. It provides technical guidance to internal software and hardware development teams and fulfills the role of an IBM extension to the open source Linux development community.

The LTC is a worldwide development team within IBM whose goal is to use world-class programming resources and software technology from IBM to actively accelerate the growth of Linux as an enterprise operating system while simultaneously helping IBM brands exploit Linux for market growth.

The LTC currently has programmers involved in many Linux projects, including scalability, serviceability, OS security, network security, networking, file systems, volume management, performance, directory services, standards, documentation, accessibility, test, security certification, systems management, cluster management, high availability, storage and I/O, PowerPC® support, power management, reliability, internationalization, and other projects required to make Linux a mature operating system that is ready for mission-critical workloads.

Members of the LTC work directly in the open source community using standard open source development methodology. They work as peers within the shared vision of the Linux community leadership and participate in setting the Linux design and development direction.

1.5 Understanding the organization of this book

We have organized this IBM Redbooks publication as follows:

- Understanding hardware subsystems
- Understanding operating system performance
- Working with performance monitoring tools
- Detecting and removing performance bottlenecks
- Tuning applications

Part 1, “2.3, “Server subsystems” on page 30” covers each of the major subsystems and their contributions to the overall performance of the server:

- CPU
- Chipsets
- PCI bus architecture
- Memory
- Disk subsystem
- Network adapter
- Operating system

Part 2, Chapter 4, “Linux” on page 67 describes performance aspects of the Linux operating system distributions that are covered in this book:

- Red Hat Enterprise Linux
- SUSE Linux Enterprise Server

Part 3, Chapter 5, “Linux monitoring tools” on page 87 covers the tools that are available to users of System p servers that run these operating systems. With these tools, it is possible to capture and analyze performance data so that you can identify and remove both existing and future performance bottlenecks.

Part 4, Chapter 6, “Identifying bottlenecks” on page 141 takes these tools to the next step by describing how to use them. It includes:

- How to spot a performance problem and solve it quickly
- A detailed explanation of the analysis of performance bottlenecks
- Case studies that show real-life examples of performance analysis

Part 5, Chapter 8, “File and printer server tuning” on page 207 describes the performance tuning approach for some of the popular server applications. such as:

- DB2 UDB
- Oracle
- WebSphere
- Apache
- Samba

2

IBM System p hardware

In this chapter, we provide an overview of the IBM System p product line, its advantages, and describe the subsystem of the server hardware.

2.1 System p advantages

The IBM System p servers are designed for greater application flexibility, with innovative technology, to capitalize on the on demand revolution in server environments. The IBM System p servers redefined the IT economics of enterprise UNIX and Linux computing, by providing outstanding performance, scalability, RAS and manageability, and the industrial leading virtualization feature.

2.1.1 Performance

IBM System p servers and BladeCenter® JS21 servers feature the latest POWER processor technology, deliver leading performance comparing to other coeval servers from various vendors, in both benchmark program and real word application.

IBM POWER is a mature and widely used processor technology. The System p5™ servers use POWER5/POWER5+™ chip based on the POWER technology and optimized for business computing.

The BladeCenter JS21 uses a PowerPC 970 processor, which is optimized for business and scientific computing. It features an extra vector processing engine named *AltiVec*. Applications with a great deal of vector computing can take advantage of the engine to dramatically speed up the process.

Table 2-1 listed the place of IBM System p servers in several business workload benchmarks. It clearly shows out how good IBM System p servers perform in business workloads.

Table 2-1 Benchmarks

Benchmark	First	Second	Third
TPC-C V5 single system	p5-595	p5-595	p5-595
TPC-H 10TB v2	p5-575		
SAP SD 2 Tier R/3	p5-595		p5-595
SAP SD 3 Tier R/3	p5-595		
Oracle Apps Std. 11i v11.5.9	p5-570		
Oracle ASB v11.5.9 Batch	p5-570		

Figure 2-1 shows IBM is the clear leader of supercomputing. IBM POWER technology based processor contributed more than one-third of the total computing power of the Top 500 supercomputer list, which far exceeds any other processor architecture.

Note: The data used in Figure 2-1 is taken from the June 2006 Top 500 supercomputer list, which is the latest result at the time the book was written. It is available at the following URL:

http://www.top500.org/lists/2006/06

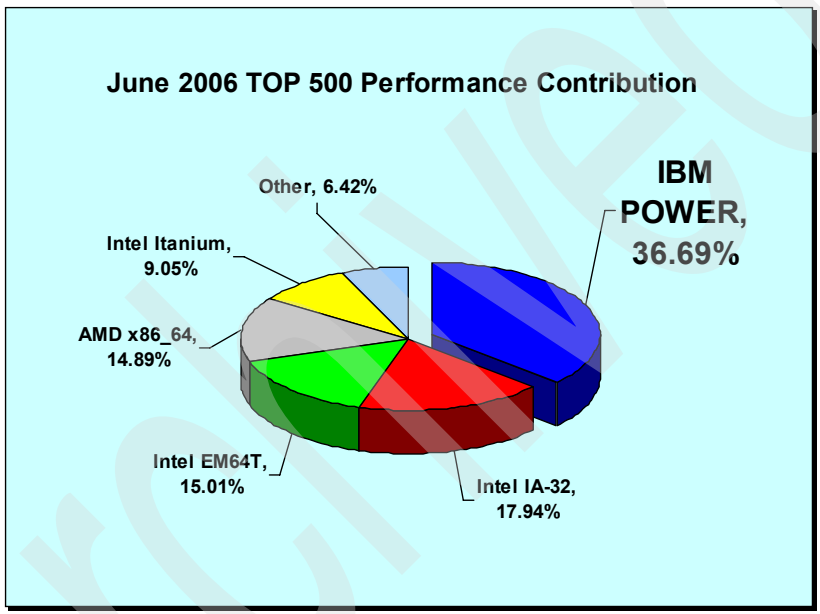


Figure 2-1 IBM POWER in TOP500

2.1.2 Scalability

Modern applications are demanding more and more computing resources, including CPU, memory, network, and so on. The server hardware has to expand to fulfill the requirement. There are two ways for hardware to scale: scale up and scale out.

Scale up

Scale up means to have more processor, memory, network, and other necessary resources in a box (they are tightly coupled to build a system unit). The unit just has more components added to gain more computing power. Typically, if the application needs more processor for computation, use an SMP system and add more CPUs.

The IBM System p product line has a full family of servers equipped with the high performance POWER5/POWER5+ processor, from 1-core, 2-core, 4-core, 8-core, 16-core to 32-core and end up with 64-core, and the memory scales from 1GB to 2 TB. The IBM System p5 595 is a typical representative of this line. It scales up to 64 processors with 2 TB of memory, which can fulfill the resource needs of an every day application. That is why p5-595 is the unbeatable leader of many performance-critical tasks.

Scale out

Scale out means have several system units process the task concurrently. While each unit is a individual box that has all the subsystems and can handle the application itself, many of the loosely connected units can form a large cluster that have the computing task distributed to each unit, by means of load balancing technology or parallel executive environment.

For the load balancing type of scale out, workloads are dispatched to the system units evenly, and each unit deals with different requests from input, so that the whole cluster can aggregate the power of all of them.

Parallel executive is usually seen in a High Performance Computing (HPC) cluster. The system units (usually called nodes) are interconnected through a high speed network, sometimes just the gigabyte Ethernet, sometimes through more advanced technology like InfiniBand. The workload is usually a long-running CPU/memory/network sensitive computing task, divided into pieces and distributed to each units. The executive is scheduled to make the process as parallel as possible. The interim data is transferred between nodes through the highly efficient network facility.

Scale out schema often requires the hardware to be rack space efficient since a cluster may have hundreds of system units. The IBM System p5 505/505Q, 510/510Q, 575, and the Blade JS20/JS21 server fit into it quite well, and they have adequate processor and memory capacity in a very compact form factor. The p5-575 and JS21 blade can even reach the unbelievable high density of 8-core per rack unit.

2.1.3 Reliability, availability, and serviceability

Excellent quality and reliability are inherent in all aspects of the IBM System p processor design and manufacturing. The fundamental objective of the design approach is to minimize outages. The RAS features help to ensure that the system operates when required, performs reliably, and efficiently handles any failures that might occur. This is achieved using capabilities that are provided by both the hardware and the operating system.

Fault avoidance

System p5 servers are built on a quality-based design that is intended to keep errors from happening. This design includes the following features:

- Reduced power consumption and cooler operating temperatures for increased reliability
- Mainframe-inspired components and technologies

First-failure data capture

The IBM System p servers incorporate an advanced capability in start-up diagnostics and in runtime first-failure data capture (FDDC) based on strategic error checkers built into the chips. Any errors detected by the pervasive error checkers are captured into Fault Isolation Registers (FIRs), which can be interrogated by the service processor.

Permanent monitoring

The service processor (SP) included in the IBM System p servers provides a way to monitor the system even when the main processor is inoperable.

- Mutual surveillance

  The SP can monitor the operation of the firmware during the boot process and it can monitor the operating system for loss of control. It also allows the operating system to monitor for service processor activity and can request a service processor repair action if necessary.

- Environmental monitoring

  Environmental monitoring that is related to power, fans, and temperature is done by the System Power Control Network (SPCN).

Self-healing

- *Bit steering* in the redundant memory in the event of a failed memory module keeps the server operational.
- *Bit-scattering* allows for error correction and continued operation in the presence of a complete chip failure (Chipkill™ recovery).
- Single bit error correction using ECC without reaching error thresholds for main, L2, and L3 cache memory.
- L3 cache line deletions have been extended from two to 10 for additional self-healing.
- ECC has been extended to inter-chip connections on fabric and processor bus.
- *Memory scrubbing* helps prevent soft-error memory faults.

N+1 redundancy

The use of redundant parts allows the p5-550 and p5-550Q to remain operational with full resources:

- Redundant spare memory bits in L1, L2, L3, and main memory
- Redundant fans
- Redundant power supplies (optional)

Fault masking

If corrections and retries succeed and do not exceed the threshold limits, the system remains operational with full resources and deferred maintenance is possible.

- CEC bus retry and recovery
- PCI-X bus recovery
- ECC Chipkill soft error

Resource deallocation

If recoverable errors exceed the threshold limits, resources can be deallocated with the system remaining operational, allowing deferred maintenance at a convenient time.

- Processor
- L3 cache
- Partial L2 cache deallocation
- Memory
- PCI-X bus and slots
- Deconfigure or bypass failing I/O adapters

Serviceability

Increasing service productivity means the system is up and running for a longer time. The IBM System p servers improve service productivity by providing the functions that are described in the following sections.

- Error indication and LED indicators

  The server provides internal LED diagnostics that identifies parts that require service.

- Concurrent Maintenance provides replacement of the following parts while the system remains running:
  - Disk drives
  - Cooling fans
  - Power subsystems
  - PCI-X adapter cards
  - Operator Panel (requires HMC guided support)
  - Dual Port I/O HUB RIO-2/HSL-2 Adapter (FC 1806)

2.1.4 Manageability

Many functions and tools are provided for IBM System p servers to ease management.

Service Processor

The service processor (SP) is always working and checking the system for errors, ensuring the connection to the HMC (if present) for manageability purposes and accepting Advanced System Management Interface (ASMI) SSL network connections. The SP provides the ability to view and manage the machine-wide settings using the ASMI and allows complete system and partition management from HMC. Also, the surveillance function of the SP is monitoring the operating system to check that it is still running and has not stalled.

Partition diagnostics

The diagnostics consist of stand-alone diagnostics, which are loaded from the DVD-ROM drive and online diagnostics (available in AIX 5L™).

Service Agent

Service Agent is an application program that operates on an IBM System p computer and monitors them for hardware errors. It reports detected errors, assuming they meet certain criteria for severity, to IBM for service with no client intervention. It is an enhanced version of Service Director™ with a graphical user interface.

Using Service Agent for System p, you can accomplish:

- Automatic VPD collection
- Automatic problem analysis
- Problem-definable threshold levels for error reporting
- Automatic problem reporting where service calls are placed to IBM without intervention
- Automatic client notification

Service Focal Point

Service Focal Point is an application on the HMC that enables you to diagnose and repair problems on the system. In addition, you can use Service Focal Point to initiate service functions on systems and logical partitions that are not associated with a particular problem. You can configure the HMC to use the Service Agent call-home feature to send IBM event information. Service Focal Point is available also in Integrated Virtualization Manager.

Concurrent firmware maintenance

The IBM System p5 client-Managed Microcode is a methodology that enables you to manage and install microcode updates on System p servers and associated I/O adapters. The IBM system p5 Microcode can be installed either from the HMC or from a running partition. Concurrent Firmware Maintenance (CFM) function on System p5 systems was introduced in system firmware level 01SF230_126_120, which was released on June 16, 2005. This function supports nondisruptive system firmware service packs to be applied to the system concurrently (without requiring a reboot to activate changes).

2.1.5 Virtualization

Virtualization is a critical component in the on demand operating environment and the system technologies implemented in the POWER5/POWER5+ processor-based IBM System p5 servers provide a significant advancement in the implementation of functions required for operating in this environment. IBM virtualization innovations on the product line provide industry-unique utilization capabilities for a more responsive, flexible, and simplified infrastructure.

The IBM System p5 family of servers includes powerful new virtualization capabilities, such as the partitioning of processors to 1/10th of a processor, sharing processor resources in a pool to drive up utilization, sharing physical disk storage and communications adapters between partitions, and taking advantage of cross-partition workload management, to mention a few of them.

Some of the reasons why virtualization technology is so important:

- Reduce costs by increasing asset utilization.
- Re-deploy talent to manage your business, not the infrastructure.
- Rapidly provision new servers.
- Drive new levels of IT staff productivity.
- Simplify server management and operations.
- Communicate more securely with virtual Ethernet.

Please refer to Chapter 3, “System p server virtualization” on page 49 for more information about System p virtualization technology.

2.2 System p product line introduction

The IBM System p server is a re-brand of the famous IBM eServer™ pSeries® server. The product line consist of rack optimized servers, deskside tower servers, and big irons. Each type of the server is optimized for different workloads, such as transaction processing, Web application hosting, file serving, scientific computing, and so on.

All the IBM System p servers use IBM POWER technology based processor: POWER5™, POWER5+, or PowerPC 970. IBM POWER architecture provides the performance and reliability advantages for a perfect server platform.

The IBM System p servers have ECC Chipkill technology protected DDR/DDR2 SDRAM DIMM, which provides a highly reliable memory subsystem. The POWER architecture has a memory controller integrated into the processor, which, along with the Synchronous Memory Interface II (SMI-II), makes the memory access both low latency and high throughput. To fit into different workloads, each server type supports a maximum memory capacity from 32 GB up to 2048 GB.

The IBM System p server disk storage supports:

- Internal Ultra320 SCSI disk drive, with a total capacity of up to several terabytes, with an optional RAID 0, 5, 10 protect
- External Remote I/O (RIO) Drawer, EXP24 SCSI enclosure
- Storage Area Network (SAN) array, Network Attached Storage (NAS), and Internet SCSI (iSCSI)

The IBM System p family of servers provide virtualization features to help clients exploit more value from their servers. It allows you to:

- Build a flexible management fabric for virtualizing resources across your IT infrastructure with services providing provisioning and workload management.
- Optimize resource usage with such technologies as dynamic logical partitioning.
- Build a common, consistent, and cross-platform systems management solution for IBM servers, storage, and operating systems.
- Better optimize resource utilization and help reduce cost and complexity while providing cost-effective IT solutions for heterogeneous environments.
- Deepen the integration of IT with business using advanced virtualization technologies and management services.

2.2.1 POWER servers

Here is a brief introduction to the available IBM System p servers. To simplify naming, they are referred to as p5-5XX or p5-5XXQ.

Note: This is only a brief summary of each server, to provide a overview of the products and their key characteristic. For more detailed information, please refer to the IBM Web site product page; the IBM Redbooks Web site also has several very good Redbooks providing such details.

p5-505 and p5-505Q

The p5-505 and p5-505Q servers come in a 1U rack package, feature one or two POWER5+ processors with one, two, or four active processor cores running at 2.1 or 1.9 GHz (1-core and 2-core) or 1.65 GHz (4-core).

The p5-505 and p5-505Q servers support a maximum of 32 GB of 533 MHz DDR2 memory. It can have up to two internal SCSI disk drives installed, with optional RAID 0 and 10 support.

For more details, please refer to *IBM System p5 505 and 505Q Technical Overview and Introduction*, REDP-4079 at:

http://www.redbooks.ibm.com/abstracts/redp4079.html?Open

Figure 2-2 shows the front angle view of the p5-505 or 505Q server.

Figure 2-2 IBM System p5 505 or 505Q server

p5-510 and p5-510Q

The p5-510 and p5-510Q servers are in a 2U rack drawer package, feature one or two POWER5+ or POWER5 processors, and have one, two, or four processor cores. Systems with one or two cores are available with 1.5 GHz, 1.9 GHz, or 2.1 GHz processors and are installed on a dual-core module (DCM). Systems with four cores are available with 1.5 GHz or 1.65GHz processors, which are installed on a quad-core module (QCM).

The p5-510 and p5-510Q servers support maximum 32GB of 533 MHz DDR2 memory. And up to four internal SCSI disk drives can be installed to have maximum 1.2 TB internal storage (using the disk drive features that are available at the time of writing), with optional RAID 0, 5or 10 support.

For more detail please refer to *IBM System p5 510 and 510Q Technical Overview and Introduction*, REDP-4136, available from:

`http://www.redbooks.ibm.com/abstracts/redp4136.html?Open`

Figure 2-3 shows the front angle view of the p5-510 or 510Q server.

Figure 2-3 IBM System p5 510 or 510Q server

p5-520 and p5-520Q

The p5-520 and p5-520Q servers come in 4U rack mount or deskside packages. The p5-520 features one or two POWER5+ processors, each with one or two cores running at 1.65 GHz, 1.9 GHz, or 2.1 GHz. The p5-520Q comes with four cores running at 1.5 GHz or 1.65 GHz.

The p5-520 and p5-520Q servers support a maximum of 32 GB of 533 MHz DDR2 memory. The internal SCSI storage can go up to 2.4 TB, with eight disk drives split into two 4-pack enclosures. RAID support can be configured by adding a SCSI RAID daughter card that plugs directly on the system board, or with a PCI-X Dual Channel Ultra320 SCSI RAID adapter to drive a 4-pack enclosure. The two configurations both support RAID 0, 5, or 10.

For more details, please refer to *IBM System p5 520 and 520Q Technical Overview and Introduction*, REDP-4137 at:

http://www.redbooks.ibm.com/abstracts/redp4137.html?Open

Figure 2-4 shows the front angle view of the p5-520 or 520Q server.

Figure 2-4 IBM System p5 520 or 520Q rack model

p5-550 and p5-550Q

The p5-550 and p5-550Q servers come in 4U rack mount or deskside packages. They can support up to two processor cards each with 2-core processor cards running at 1.65 GHz, 1.9 GHz, or 2.1 GHz (DCM card), or each with two 4-core processor cards running at 1.5 GHz or 1.65 GHz (QCM card).

The p5-550 and p5-550Q servers supports a maximum of 64 GB of 533 MHz DDR2 memory when configured with two processor cards. The internal SCSI storage can go up to 2.4 TB, with eight disk drives split into two 4-pack enclosures. RAID support can be configured by adding a SCSI RAID daughter card that plugs directly on the system board, or with a PCI-X Dual Channel Ultra320 SCSI RAID adapter to drive a 4-pack enclosure. The two configuration both support RAID 0, 5, or 10.

For more details, please refer to *IBM System p5 550 and 550Q Technical Overview and Introduction*, REDP-4138 at:

http://www.redbooks.ibm.com/abstracts/redp4138.html?Open

Figure 2-5 shows the front angle view of the p5-550 or 550Q deskside model, with the front cover removed.

Figure 2-5 IBM System p5 550 or 550Q deskside model

p5-560Q

The p5-560Q rack-mount server is a 4U symmetric multiprocessor (SMP) system, based on a building block architecture and Quad-Core Module (QCM) technology. The system features up to four 4-core processor cards with POWER5+ processors running at 1.5 GHz and each building block can have up to two processor cards.

The p5-560Q server supports a maximum of 128 GB of 533 MHz DDR2 memory when configured with two building blocks. Each building block is equipped with one 6-pack disk drive enclosure, so the maximum internal storage capacity for a combined system made of two building blocks is 3.6 TB (using the disk option available at the time of writing).

For more details, please refer to *IBM System p5 560Q Technical Overview and Introduction*, REDP-4139 at:

http://www.redbooks.ibm.com/abstracts/redp4139.html?Open

Figure 2-6 shows the front angle view of a p5-560Q server with two building blocks stacked in a rack.

Figure 2-6 IBM System p5 560Q server, two building block

p5-570

The p5-570 server rack-mount server is a 4U building block that can be made of one to four building blocks. Each of the building blocks can contain up to two 2-core processor cards with POWER5+ processors running at 1.9 GHz or 2.2 GHz. The whole system can have up to 16 POWER+ processor cores.

The maximum memory for a system consists of four p5-570 building blocks that can be up to 256 GB of 533 MHz or 512 GB of 400 MHz DDR2. Each of the p5-570 building blocks feature six disk drive bays, so a 4-building block system can have internal disk storage as large as 7.2 TB (using the disk option available at the time of writing).

For more details, please refer to *IBM System p5 570 Technical Overview and Introduction*, REDP-9117 at:

http://www.redbooks.ibm.com/redpieces/abstracts/redp9117.html?Open

Figure 2-7 shows the front angle view of the p5-570 server.

Figure 2-7 IBM System p5 570 server

p5-575

The p5-575 system is in a 2U rack-mount form factor and designed to be the computing node of a High Performance Computing (HPC) cluster. Each of the nodes can contain 8-core POWER5+ processor cards running at 2.2 GHz or 16-core processor cards running at 2.2 GHz in a 2U rack-mount drawer.

The p5-575 server supports a maximum of 256 GB of 533 MHz DDR2. Up to two SCSI disk drives can be installed to have maximum of 600 GB internal storage (using the disk option available at the time of writing).

Figure 2-8 shows the front angle view of the p5-575 server, with the cover opened.

Figure 2-8 IBM System p5 575 server, with cover opened

p5-590 and p5-595

The p5-595 server is the flagship of the product line. It can feature up to 64 POWER5+ processor cores running at 2.1 GHz or 2.3 GHz, or the POWER5 cores running at 1.65 GHz or 1.9 GHz. Accompanying the p5-595 is the up to 32-core p5-590, with the POWER5+ processor running at 2.1 GHz or the POWER5 processor running at 1.65 GHz.

The p5-595 supports a maximum of 2048 GB DDR1 or DDR2 memory, and the p5-590 supports a maximum of 1024 GB DDR1 or DDR2 memory. Both of them require a minimum of 8 GB of configurable system memory.

The internal storage of p5-590 and p5-595 are supported by using I/O drawers.

For more details, please refer to *IBM System p5 590 and 595 Technical Overview and Introduction*, REDP-4024 at:

http://www.redbooks.ibm.com/redpieces/abstracts/redp4024.html?Open

Figure 2-9 shows the front view of the p5-590 or 595 server.

Figure 2-9 IBM System p5 590 or 595 server

2.2.2 PowerPC servers and blades

Here we discuss PowerPC servers and blades.

p5-185

The p5-185 server comes in a deskside or a 5U rack form factor. It is available in a 1-core or 2-core configuration, using the PowerPC 970 processor running at 2.5 GHz.

The p5-185 system supports up to 8 GB of 333 MHz DDR memory. The maximum internal storage capacity can go up to 900 GB by using three 300 GB Ultra320 disk drives (using the disk options available at the time of writing).

For more details, please refer to *IBM System p5 185 Technical Overview and Introduction*, REDP-4141 at:

http://www.redbooks.ibm.com/abstracts/redp4141.html?Open

Figure 2-10 shows the front angle view of the p5-185 server.

Figure 2-10 IBM System p5 185 rack-mount model

BladeCenter JS21

The JS21 is the high-density blade server that is compatible with IBM BladeCenter, BladeCenter H, and BladeCenter T. The Blade JS21 features two single-core PowerPC 970MP processors running at 2.7 GHz or two dual-core processors at 2.5 GHz.[1]

The Blade JS21 server supports up to 16 GB DDR2 memory. The maximum internal storage capacity is 146 GB, with up to two Serial Attached SCSI (SAS) disk drives, with RAID 0 or 10 support.

For more details, please refer to *IBM BladeCenter JS21 Technical Overview and Introduction*, REDP-4130 at:

http://www.redbooks.ibm.com/abstracts/redp4130.html?Open

[1] These are the processor frequencies when installed in BladeCenter H. The frequency is reduced to 2.6 GHz (single-core) and 2.3 GHz (dual-core) when installed in other BladeCenter chassis.

Figure 2-11 shows the front angle view of the JS21 Blade server.

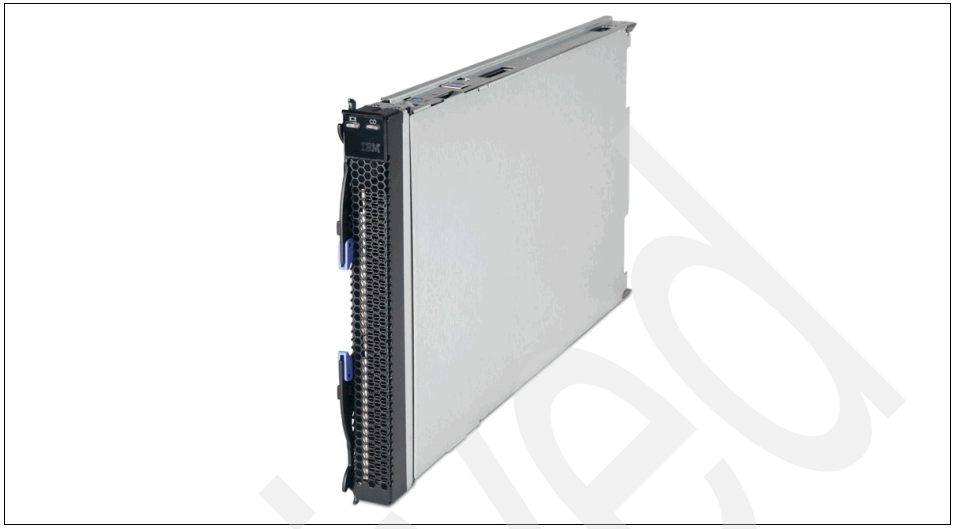


Figure 2-11 IBM Blade JS21

2.2.3 IntelliStation

Here we discuss the IntelliStation® workstations.

IntelliStation 185

The IntelliStation POWER 185 workstation comes in a deskside package and is available with either a 1-core or 2-core PowerPC 970 processor running at 2.5 GHz.

The IntelliStation 185 supports a maximum of 8 GB of 333 MHz DDR memory. It features up to three disk drive bays, with a maximum 900 GB of internal storage capacity (using the disk options available at the time of writing).

For more details, please refer to *IBM IntelliStation POWER 185 Technical Overview and Introduction*, REDP-4135 at:

`http://www.redbooks.ibm.com/abstracts/redp4135.html?Open`

Figure 2-12 shows the front angle view of the IntelliStation POWER 185.

Figure 2-12 IBM IntelliStation POWER 185

IntelliStation 285

The IntelliStation POWER 285 comes in a deskside package and is available in a 1-core or 2-core processor configuration using POWER+ processors running at 1.9 GHz or 2.1 GHz.

The IntelliStation 285 features a maximum of 32 GB of 533 MHz DDR2 memory. The configuration include a 4-pack disk drive enclosure, which supports a maximum of 1.2 TB of internal storage capacity (using the disk options available at the time of writing).

For more details, please refer to *IBM IntelliStation POWER 285 Technical Overview and Introduction*, REDP-4078 at:

http://www.redbooks.ibm.com/abstracts/redp4078.html?Open

Figure 2-13 shows the front angle view of the IntelliStation POWER 285.

Figure 2-13 IBM IntelliStation POWER 285

2.2.4 Others

Here we discuss other workstations.

Hardware Management Console

The Hardware Management Console (HMC) is a dedicated desktop PC workstation that provides several functions for configuring and operating IBM System p servers and IBM eServer pSeries servers. The HMC connects to the managed systems through Ethernet to send the directions to the Service Processor.

The HMC is an important component of the virtualization environment. It provides the following functions:

- Logical partitioning management:
  - Starting, stopping, resetting, and shutting down a partition.
  - Opening a virtual console for each partition or connected pSeries server system.
  - Creating partition profiles that define the processor, memory, and I/O resources allocated to an individual partition.
  - Performing DLPAR operations that dynamically change the resource allocation (such as processor, memory, and I/O) for the specified partition.
- Displaying system resources and status.

- Booting, starting, and stopping the connected System p server systems.
- Configuring the HMC itself.
- Managing the HMC software level.
- A service focal point that gives you the tools for problem determination and service support, such as call-home and error log notification through an analog phone line.

The HMC comes in a 1U rack-mount or deskside PC package. It features one DVD-RAM drive, one Ethernet port, two native serial ports, six USB ports, as well as other ports.

For more details, please refer to *Effective System Management Using the IBM Hardware Management Console for pSeries*, SG24-7038 at:

http://www.redbooks.ibm.com/abstracts/sg247038.html?Open

7311 model D20 Remote I/O Drawer

The 7311 model D20 Remote I/O Drawer is a 4U rack-mount enclosure and has the following attributes:

- 7 PCI-X slots 3.3 volt, keyed, 133 MHz hot-pluggable
- Two 6-pack hot-swappable SCSI devices
- Optional redundant hot-swap power
- Two RIO-2 ports and two SPCN ports

When IBM System p servers need more PCI-X slots, 7311-D20 drawers can be attached, which makes them especially well-suited to extend the number of partitions. Depending on the system type and configuration, the maximum number of I/O drawers supported is different.

Figure 2-14 shows the front view of 7311-D20 RIO Drawer.

Figure 2-14 IBM System p 7311-D20 Remote I/O Drawer, front view

IBM TotalStorage EXP24 Expandable Storage

The IBM TotalStorage® EXP24 Expandable Storage disk enclosure, Model D24 or T24, provides low-cost Ultra320 (LVD) SCSI disk storage. This disk storage enclosure device provides more than 7 TB of disk storage in a 4U rack-mount (Model D24) or compact deskside (Model T24) unit. It provides 24 hot-swappable disk bays, 12 accessible from the front and 12 from the rear, separated into four 6-pack disk drive enclosures. Each of the enclosures can be attached independently to an Ultra320 SCSI or RAID adapter. For highly available configurations, a dual bus repeater card allows each 6-pack to be attached to two SCSI adapters, installed in one or multiple servers or logical partitions. Optionally, the two front or two rear 6-packs can be connected together to form a single Ultra320 SCSI bus of 12 drives.

Figure 2-15 shows the front angle view of EXP24 enclosure.

Figure 2-15 IBM TotalStorage EXP24

2.3 Server subsystems

Servers are made up of a number of subsystems, each of which plays an important role in how the server performs. The major subsystems in a server are:

- Processor
- Memory
- Disk
- Network

The subsystems that are critical to performance depend on the workload type running on the server. The bottlenecks that occur can be resolved by gathering and analyzing performance data. Chapter 5, “Linux monitoring tools” on page 87 introduce the tools that can be used to monitor the subsystems.

2.3.1 Processor subsystem

Here we discuss the processor subsystem.

The POWER5 processor

The POWER5 processor features single-threaded and multi-threaded execution, providing higher performance in the single-threaded mode than its POWER4™ predecessor provides at equivalent frequencies. The POWER5 microprocessor maintains both binary and architectural compatibility with existing POWER4 systems to ensure that binaries continue executing properly and that all application optimizations carry forward to newer systems. The POWER5 microprocessor provides additional enhancements, such as virtualization, simultaneous multi-threading support, and improved reliability, availability, and serviceability at both chip and system levels, and it has been designed to support interconnection of 64 processors along with higher clock speeds.

Figure 2-16 on page 31 shows the high-level structures of POWER4 and POWER5 processor-based systems. The POWER4 processors scale up to a 32-core symmetric multi-processor. Going beyond 32 processors with POWER4 architecture could increase interprocessor communication, resulting in higher traffic on the interconnection fabric bus. This can cause greater contention and negatively affect system scalability.

Moving the L3 cache reduces traffic on the fabric bus and enables POWER5 processor-based systems to scale to higher levels of symmetric multi-processing. The POWER5 processor supports a 1.9 MB on-chip L2 cache, implemented as three identical slices with separate controllers for each. Either processor core can independently access each L2 controller. The L3 cache, with a capacity of 36 MB, operates as a backdoor with separate buses for reads and writes that operate at half the processor speed.

Because of the higher transistor density of the POWER5 0.13-µm technology (over the original POWER4), it was possible to move the memory controller on-chip and eliminate a chip that was previously needed for the memory controller function. These changes in the POWER5 processor also have the significant side benefits of reducing latency to the L3 cache and main memory, as well as reducing the number of chips that are necessary to build a system.

The POWER5 processor supports the 64-bit PowerPC architecture. A single die contains two identical processor cores, each supporting two logical threads. This architecture makes the chip appear as a 4-way symmetric multi-processor to the operating system. The POWER5 processor core has been designed to support both enhanced simultaneous multi-threading and single-threaded (ST) operation modes.

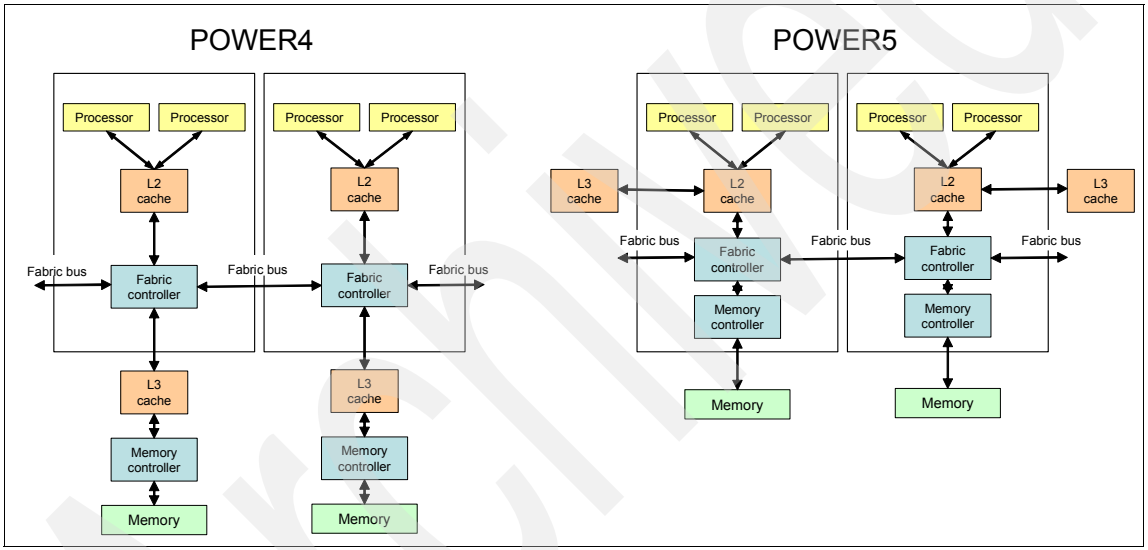


Figure 2-16 POWER4 and POWER5 system structures

The POWER5+ chip capitalizes on all the enhancements brought by the POWER5 chip.

Figure 2-17 shows a high-level view of the POWER5+ chip.

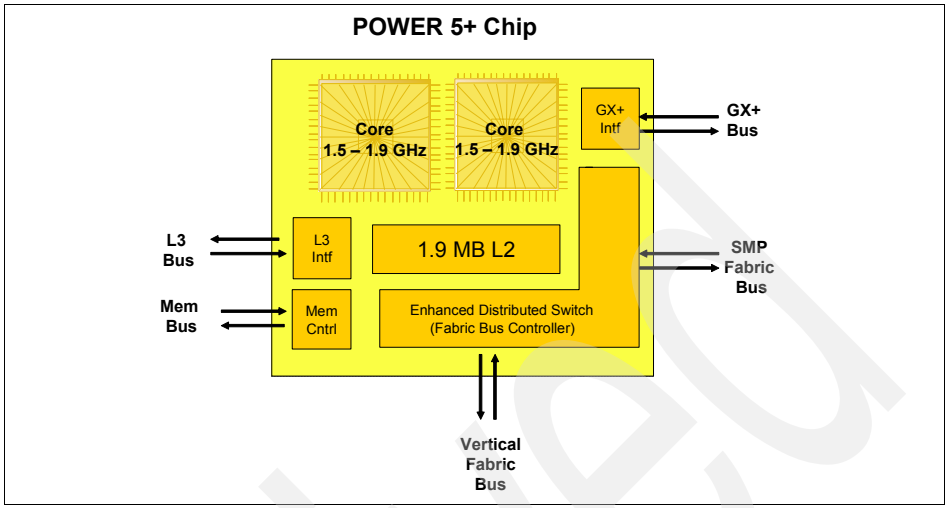


Figure 2-17 POWER5+ chip

The CMOS10S technology in the POWER5+ chip uses a 90 nm fabrication process, which enables:

- Performance gains through faster clock rates
- Chip size reduction (243 mm compared with 389 mm)

The POWER5+ chip is 37% smaller than the POWER5 chip. It consumes less power and requires less sophisticated cooling. Thus, you can use the POWER5+ chip in servers where previously you could only use low frequency chips due to cooling restrictions.

The POWER5+ design provides the following additional enhancements:

- New pages sizes in ERAT and TLB. Two new pages sizes (64 KB and 16 GB) were recently added in the PowerPC architecture.
- New segment size in SLB. One new segment size (1 TB) was recently added in the PowerPC architecture.
- The TLB size has been doubled in the POWER5+ over the POWER5 processor. The TLB in POWER5+ has 2048 entries.
- Floating-point round to integer instructions. New instructions (frfin, frfiz, frfip, and frfim) have been added to round floating-point numbers integers with the following rounding modes: nearest, zero, integer plus, and integer minus.
- Improved floating-point performance.
- Lock performance enhancement.

- Enhanced SLB read.
- Support for the True Little-Endian mode as defined in the PowerPC architecture.
- Double the SMP support. Changes have been made in the fabric, L2 and L3 controller, memory controller, GX+ controller, and chip RAS to provide support for the Quad-Core Module (QCM) that allows the SMP system sizes to be double what is available in POWER5 DCM-based servers. However, the current POWER5+ implementations support only a single address loop.
- Several enhancements have been made in the memory controller for improved performance. The POWER5+ is ready to support DDR2 667 MHz DIMMs in the future.
- Enhanced redundancy in L1 Dcache, L2 cache, and L3 directory. Independent control of the L2 cache and the L3 directory for redundancy to allow split-repair action has been added. More wordline redundancy has been added in the L1 Dcache. In addition, Array Built-In Self Test (ABIST) column repair for the L2 cache and the L3 directory has been added.

POWER5+ Quad-Core Module

There is now the POWER5+ Quad-Core Module (QCM) and the local memory storage subsystem for that QCM. Two POWER5+ dual core chips and their associated L3 cache chips are packaged in the QCM.

Figure 2-18 shows a layout view of a QCM with the associated memory.

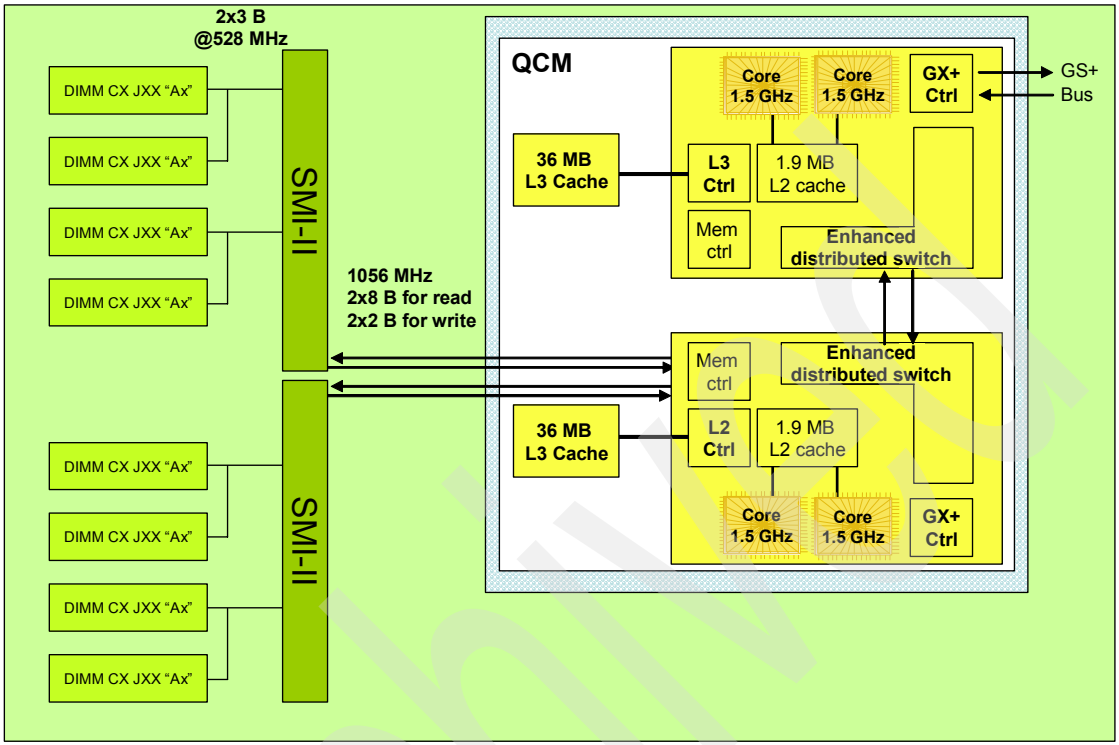


Figure 2-18 POWER5+ 1.5 GHz processor card with DDR2 memory socket layout view

The storage structure for the POWER5+ chip is a distributed memory architecture that provides high-memory bandwidth. Each processor in the QCM can address all memory and sees a single shared memory resource. In the QCM, one POWER5+ chip has direct access to eight memory slots, controlled by two SMI-II chips, which are located in close physical proximity to the processor modules. The other POWER5+ chip has access to the same memory slots through the Vertical Fabric Bus.

Both POWER5+ dual core chips in the QCM access their own 36 MB L3 cache for a total of 72 MB L3 cache in the QCM. The theoretical maximum throughput of the L3 cache is 16 byte read, 16 byte write, at a bus frequency of 750 MHz (based on a 1.5 GHz processor clock), which equates to 24000 MBps or 24 GBps.

I/O connects to the QCM using the GX+ bus. The QCM provides a single GX+ bus. Each processor in the POWER5+ chips has either a direct access to the GX+ Bus using its GX+ Bus controller or uses the Vertical Fabric Bus controlled by the Fabric Bus controller. The GX+ bus provides an interface to I/O devices through the RIO-2 connections.

The POWER5+ processor that does not have direct access to memory, but it does have direct access to the GX+ Bus, as shown in Figure 2-19.

In the two processor cards configuration (8-core p5-550Q, for example), the QCM to QCM communication is implemented using the Horizontal Fabric Bus.

Figure 2-19 shows a high-level layout of a two processor card p5-550Q configuration.

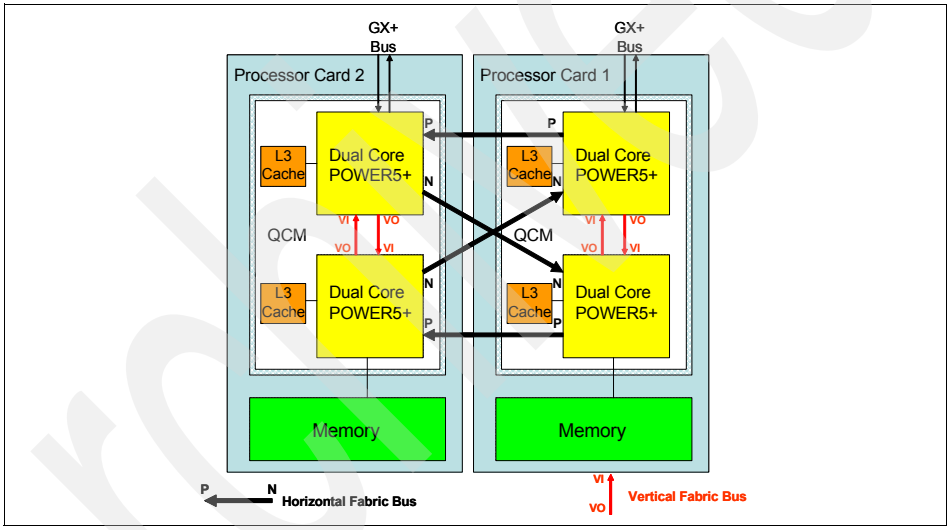


Figure 2-19 IBM System p5 550Q 8-core processor configuration

POWER5+ Dual-Core Module

The 2-core p5-550 processor card contains a Dual-Core Module (DCM) and the local memory storage subsystem for that DCM. The POWER5+ dual core chip and its associated L3 cache chip are packaged in the DCM.

Figure 2-20 shows a layout view of DCM and associated memory.

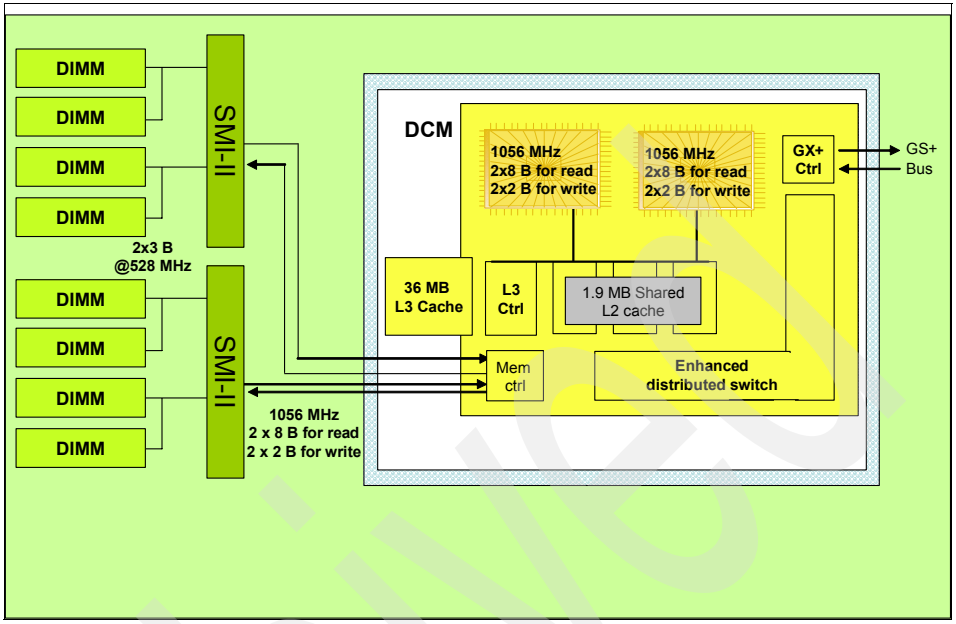


Figure 2-20 POWER5+ 1.9 GHz DCM with DDR2 memory socket layout view

The storage structure for the POWER5+ chip is a distributed memory architecture that provides high-memory bandwidth, although each processor can address all the memory and sees a single shared memory resource. They are interfaced to eight memory slots, controlled by two SMI-II chips, which are located in close physical proximity to the processor modules.

The I/O connects to the p5-550 processor module using the GX+ bus. The processor module provides a single GX+ bus. The GX+ bus provides an interface to I/O devices through the RIO-2 connections.

The theoretical maximum throughput of the L3 cache is 16 byte read and 16 byte write at a bus frequency of 950 MHz (based on a 1.9 GHz processor clock), which equates to 30400 MBps or 30.40 GBps.

Simultaneous multi-threading (SMT)

As a permanent requirement for performance improvements at the application level, simultaneous multi-threading functionality is embedded in the POWER5 chip technology. Developers are familiar with process-level parallelism (multi-tasking) and thread-level parallelism (multi-threads). Simultaneous multi-threading is the next stage for achieving higher processor utilization for throughput-oriented applications to introduce the method of instruction group-level parallelism to support multiple pipelines to the processor. The instruction groups are chosen from different hardware threads belonging to a single OS image.

Simultaneous multi-threading is activated by default for supported Linux distributions. On a 2-core POWER5 processor-based system, the operating system discovers the available processors as a 4-core system. To achieve a higher performance level, simultaneous multi-threading is also applicable in Micro-Partitioning™, capped or uncapped, and dedicated partition environments.

The simultaneous multi-threading mode increases the usage of the execution units. In the POWER5 chip, more rename registers have been introduced (both Floating-Point registers (FPR) and general-purpose registers (GPR) are increased to 120) that are essential for out-of-order execution and vital for the simultaneous multi-threading.

Figure 10-5 on page 322 shows the performance gain of SMT in our test scenario.

Enhanced simultaneous multi-threading features

To improve simultaneous multi-threading performance for various workload mixes and provide robust quality of service, POWER5 provides two features:

- Dynamic resource balancing
  - The objective of dynamic resource balancing is to ensure that the two threads executing on the same processor flow smoothly through the system.
  - Depending on the situation, the POWER5 processor resource balancing logic has a different thread throttling mechanism.
- Adjustable thread priority
  - Adjustable thread priority lets software determine when one thread should have a greater (or lesser) share of execution resources.
  - The POWER5 processor supports eight software-controlled priority levels for each thread.

Single threaded operation

Not all applications benefit from simultaneous multi-threading. Having threads executing on the same processor does not increase the performance of processor intensive applications or applications that consume all of the chip's memory bandwidth. For this reason, the POWER5 processor supports the single thread (ST) execution mode. In this mode, the POWER5 processor gives all of the physical resources to the active thread, enabling it to achieve higher performance than a POWER4 processor-based system at equivalent frequencies. Highly optimized scientific codes are one example where ST operation is ideal.

Simultaneous multi-threading and ST operation modes can be dynamically switched without affecting server operations. The two modes can coexist on a single physical system; however, only a single mode is possible on each OS instance (partition).

Dynamic power management

In current CMOS[2] technologies, chip power consumption is one of the most important design parameters. With the introduction of simultaneous multi-threading, more instructions execute per cycle per processor core, thus increasing the core's and the chip's total switching power. To reduce switching power, POWER5 chips extensively use a fine-grained, dynamic clock-gating mechanism. This mechanism gates off clocks to a local clock buffer if dynamic power management logic knows that the set of latches that are driven by the buffer will not be used in the next cycle. This allows substantial power saving with no performance impact. In every cycle, the dynamic power management logic determines whether a local clock buffer that drives a set of latches can be clock-gated in the next cycle.

In addition to the switching power, leakage power has become a performance limiter. To reduce leakage power, the POWER5 chip uses transistors with low threshold voltage only in critical paths. The POWER5 chip also has a low-power mode, enabled when the system software instructs the hardware to execute both threads at priority 1. In low power mode, instructions dispatch once every 32 cycles at most, further reducing switching power. Both threads are set to priority 1 by the operating system when in the idle loop.

[2] Complementary Metal Oxide Semiconductor

2.3.2 Memory subsystem

The IBM System p servers offers pluggable DDR2 DIMMs for memory. DDR2 DIMM has a double rate compared with DDR DIMM (a DDR DIMM has double rate bits compared to SDRM) so it has up to four times the performance of traditional SDRAM. Each installed processor card provides eight slots for up to eight pluggable DDR2 DIMMs.

Figure 2-21 and Figure 2-22 on page 40 shows the memory slot and location codes for a DCM or QCM card. All of memory is accessed by two Synchronous Memory Interface (SMI)-II chips that are located between memory and processor. The SMI-II supports multiple data flow modes.

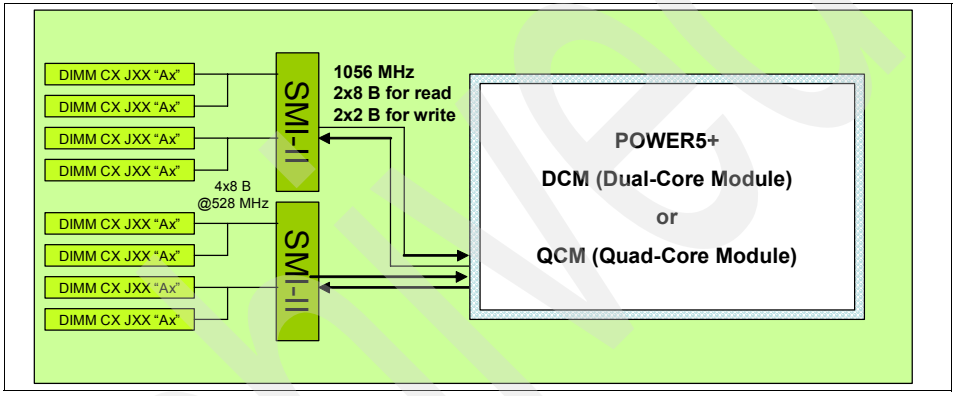


Figure 2-21 Memory subsystem

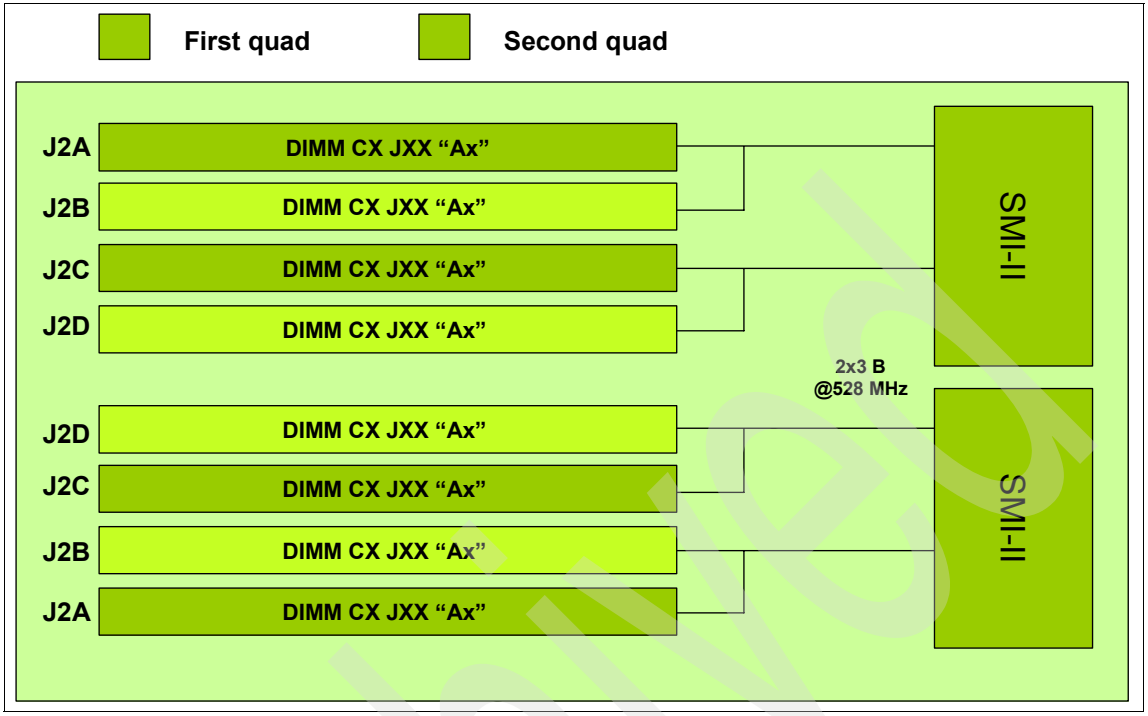


Figure 2-22 Memory location codes

Note: Figure 2-22 is just showing p5-550 and 550Q. For other server types, such as p5-595, the structure may be vastly different. Please refer to their specific document for accurate information. A good resource is the IBM Redbooks “Technical Overview” series at the following Web site:

`http://www.redbooks.ibm.com/`

Memory throughput

The memory subsystem throughput is based on the speed of the memory. An elastic interface, contained in the POWER5 chip, buffers reads and writes to and from memory and the processor. There are two Synchronous Memory Interface (SMI-II) chips, each with a single 8-byte read and 2-byte write high speed Elastic Interface-II bus to the memory controller. The bus allows double reads or writes per clock cycle. Because the bus operates at 1056 MHz, the peak processor-to-memory throughput for read is (8 x 2 x 1056) = 16896 MBps or 16.89 GBps. The peak processor-to-memory throughput for write is (2 x 2 x 1056) = 42264 MBps or 4.22 GBps, making up a total of 21.12 GBps.

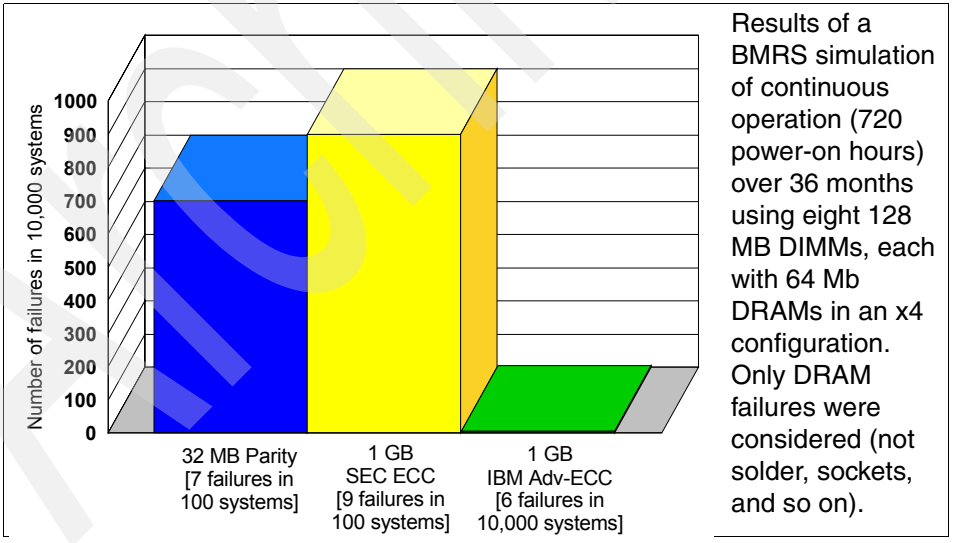
The DIMMs are 533 MHz DDR2 operating at 528 MHz through four 8-byte paths. A pair of DIMMs share one path. Read and write operations also share these paths. There must be at least four DIMMs installed to effectively use each path. In this case, the throughput between the SMI-II and the DIMMs is (8 x 4 x 528) or 16.89 GBps.

Advanced ECC memory (Chipkill)

Standard ECC (error checking and correcting) memory will detect and correct any single-bit error. It can also detect double-bit errors, but is unable to correct them. Triple-bit and larger errors may not be detected.

IBM has developed a new technology colloquially known as “Chipkill Protect ECC DIMMs”, which will allow an entire DRAM chip on a DIMM to fail while the system continues to function. These new DIMMs have been carefully designed so that there is no performance degradation over single-error correct (SEC) or standard ECC DIMMs.

Figure 2-23 shows the results of a failure rate simulation for 32 MB of parity memory, 1 GB of standard ECC (single-error correct), and 1 GB of IBM Advanced ECC memory. The simulation was for three years of continuous operation and showed the significant reduction in failures when using Advanced ECC (approximately two orders of magnitude).



Results of a BMRS simulation of continuous operation (720 power-on hours) over 36 months using eight 128 MB DIMMs, each with 64 Mb DRAMs in an x4 configuration. Only DRAM failures were considered (not solder, sockets, and so on).

Figure 2-23 Memory failure rate comparison

The capability that the data shows for the memory subsystem is fundamentally the same as the RAID technology used for the disk subsystem today. In fact, from a marketing perspective, it could be called RAID-M for Redundant Array of Inexpensive DRAMs for main Memory. This name captures the essence of its function: on-the-fly, automatic data recovery for an entire DRAM failure.

2.3.3 Disk subsystem

Here we discuss the disk subsystem.

SCSI

The most widely used drive technology in servers today is Small Computer System Interface (SCSI). There are two basic types of SCSI controller designs: array and non-array. A standard non-array SCSI controller allows the connection of SCSI disk drives to the PCI bus. Each drive is presented to the operating system as an individual, physical drive.

The array controller, a more advanced design, contains hardware designed to combine multiple SCSI disk drives into a larger single logical drive. Combining multiple SCSI drives into a larger logical drive greatly improves I/O performance compared to single-drive performance. Most array controllers employ fault-tolerant techniques to protect valuable data in the event of a disk drive failure. Array controllers are installed in almost all servers because of these advantages.

IBM System p servers use Ultra320 SCSI controllers and disk drives to provide the internal storage subsystem and the optional RAID function.

Serial Attached SCSI

Serial Attached SCSI (SAS) is the logical evolution of SCSI. SAS uses much smaller interconnects than SCSI, while offering SCSI compatibility, reliability, performance, and manageability. In addition, SAS offers longer cabling distances, smaller form factors, and greater addressability.

SAS-based products will differ from SCSI-based products in the following ways:

- Greater drive support

  SCSI-based products support 14 drives per channel. By cascading drive enclosures, SAS-based products will support up to 72 drives per four ports.

- Higher bandwidth

  Ultra320 SCSI supports 320 MBps of bandwidth per channel. SAS 1.0 supports 3 Gbps (approximately 300 MBps) of bandwidth per port. So while a PCI-X SCSI adapter with two Ultra320 SCSI channels can potentially support

640 MBps of bandwidth, a SAS-based controller with eight ports could potentially support 24 Gbps (approximately 2.4 GBps) of bandwidth. Therefore, bandwidth will be limited by PCI-X or PCI Express bus speeds.

The IBM Blade JS21 uses a SAS controller and disk drive.

Fibre Channel

Fibre Channel provides low latency and high throughput capabilities. As a result, Fibre Channel is rapidly becoming the next-generation I/O technology used to connect servers and high-speed storage. Fibre Channel addresses many of the shortcomings of SCSI with improvement in the following areas:

- Cable distance
- Bandwidth
- Reliability
- Scalability

The parallel cable used for Ultra320 SCSI limits cable distances to 25 meters or shorter. This is due to electromagnetic effects impacting signal integrity as cable length increases. Parallel cables, such as the type used by SCSI, tend to have signal interference problems because of electromagnetic coupling that occurs between parallel signals traversing the wires. Fiber optic technology allows storage to be located a maximum distance of up to 10 kilometers away from the attached server.

A significant advantage of Fibre Channel is its ability to connect redundant paths between storage and one or more servers. Redundant Fibre Channel paths improve server availability because cable or connector failures do not cause server down time, and storage can be accessed through a redundant path. In addition, both Fibre Channel and SCSI throughput can scale by utilizing multiple channels or buses between the servers and storage.

In addition to a simpler cable scheme, Fibre Channel offers improved scalability due to several very flexible connection topologies. Basic point-to-point connections can be made between a server and storage devices providing a low-cost, simple, and stand-alone connection. Fibre Channel can also be used in both loop and switch topologies. These topologies increase server-to-storage connection flexibility. The Fibre Channel loop allows up to 127 devices to be configured to share the same Fibre Channel connection. A device can be a server, storage subsystem, drive enclosure, or disk. Fibre Channel switch topologies provide the most flexible configuration scheme by theoretically providing the connection of up to 16 million devices!

IBM System p servers and Blade JS20 and JS21 could make use of a Fibre Channel array by having a Fibre Channel Adapter installed. The adapter is usually called Host Bus Adapter (HBA).

iSCSI

iSCSI is an industry standard that allows SCSI block I/O protocols (commands, sequences, and attributes) to be sent over a network using the TCP/IP protocol. This is analogous to the way SCSI commands are already mapped to parallel SCSI and Fibre Channel.

iSCSI is a network transport protocol for SCSI that operates on top of TCP. It encapsulates SCSI protocols into a TCP/IP frame, so that storage controllers can be attached to IP networks.

Unlike Fibre Channel, iSCSI uses the existing Gigabit Ethernet LAN as a medium to transfer data from the iSCSI appliance, known as the *target*, to the file or application server. At the server end, either a software iSCSI driver or a dedicated iSCSI adapter can be used to encapsulate the iSCSI blocks. This is know as the *initiator*. If a software initiator is used, then the iSCSI traffic will be transmitted through the existing network adapters. If a hardware initiator is installed, then it will need its own Ethernet network connection.

To some extent, iSCSI is limited by the speed of 1 Gb Ethernet, and the high latency of the TCP/IP protocol is the main reason for the lower throughput of iSCSI when compared with Fibre Channel. However, iSCSI is a new technology and the performance will no doubt improve as it develops.

2.3.4 Network subsystem

The network adapter is the pathway into the server. All requests to the server and all responses from the server must pass through the network adapter. Its performance is key for many server applications. The IBM System p servers are all using a Gigabyte Ethernet Network Interface Card (NIC) and most models have two built-in Gigabyte NIC ports.

Gigabyte Ethernet adapter

Different server models use different chips for the NIC. Most System p5 servers use the chip from Intel® (e1000 as the driver module in Linux), and Blade JS20 or JS21 use the chip from Broadcom (tg3 or bcm5700 as the driver module in Linux).

Different NIC chips combined with their drivers have different levels of functions provided. It may or may not provide some features that could be important for performance, like Jumbo Frame, Checksum Offloading, TOE (TCP Offload Engine), and so on.

10 Gigabit Ethernet adapters

10 Gigabit Ethernet adapters are the latest iteration of network adapters that deliver increased network throughput. They follow the IEEE 802.3ae standards and vary slightly from previous Ethernet adapters as they only operate in full duplex mode, making collision-detection protocols unnecessary. Initial adapters will also only function over optical fiber.

10 Gigabit Ethernet is capable of operating in excess of 2.5 GBps in burst mode. This, of course, presumes that the other subsystems are capable of supporting that throughput; at this rate, the current PCI bus, front-side bus, and memory bus, as well current server CPUs, would be saturated.

Although the standard has been defined for 10 Gigabit Ethernet, adapters are available from some vendors, but not commonly implemented. Also, the user needs an appropriate infrastructure to exploit it.

Virtual Ethernet adapter

One of the benefits that IBM virtualization technology provides is the Virtual Ethernet. On any IBM System p server or POWER Blades with virtualization feature, a “virtual” NIC could be created, that is, no physical adapter is actually there.

The virtual NIC could be used to communicate with the partitions that connected to the same Virtual Ethernet. With network bridging or routing properly set up, the Virtual Ethernet could be able to communicate with the outside “real” network.

The Virtual Ethernet provides up to near 10Gbps throughput under certain traffic workload. Thus, it may be quite useful for inter-partition communication: high performance with low load.

Note: More information regarding the function and performance of Virtual Ethernet is available in Chapter 3, “System p server virtualization” on page 49 and in the following IBM Redbooks:

Advanced POWER Virtualization on IBM System p5: Introduction and Configuration, SG24-7940

Advanced POWER Virtualization on IBM eServer p5 Servers: Architecture and Performance Considerations, SG24-5768

Tip: TCP offload engine (TOE)

Processing TCP/IP traffic can consume significant network, memory, CPU, and front-side bus resources. The TOE technique reduces the number of data moves across the front-side bus, effectively halving the amount of data being transferred.

When processing TCP/IP requests, the CPU is involved in the following activities:

- Packet processing
- Data movement
- Context switching
- Interrupt processing

TOE adapters eliminate packet processing from the server CPU because there is a dedicated TCP/IP engine present on the NIC. This frees up CPU for small blocks, but there is still considerable load from interrupt processing and context switching on the CPU. For large blocks, the data movement is far more efficient because a single copy is completely eliminated.

This technology will decrease the workload that the CPU and front-side bus need to do, thereby enabling the server to dedicate potentially strained resources to other tasks. 10 Gigabit adapters are capable of flooding the memory and front-side bus and using a TOE adapter in this instance will help to reduce the impact on these bottlenecks.

Infiniband

InfiniBand is a switch-based serial I/O interconnect architecture operating at a base speed of 2.5 Gbps or 10 Gbps in each direction (per port). Unlike shared bus architectures, InfiniBand is a low pin count serial architecture and enables Bandwidth Out of the Box, spanning distances up to 17 m over ordinary twisted pair copper wires. Over common fiber cable, it can span distances of several kilometers or more. Furthermore, InfiniBand provides both Quality of Service (QoS) and RAS. These RAS capabilities have been designed into the InfiniBand Architecture from the beginning and are critical to its ability to serve as the common I/O infrastructure for the next generation of compute server and storage systems at the heart of the internet. As a result, InfiniBand will radically alter the systems and interconnects of the internet infrastructure.

The InfiniBand System Fabric (shown in Figure 2-24) is backed by top companies in the industry, including all of the major server vendors. The InfiniBand Architecture offers all of the benefits mentioned, but, to realize the full performance bandwidth of the current 10 Gbps links, the PCI limitation must be removed, and this is where currently developing interconnect technologies assist InfiniBand.

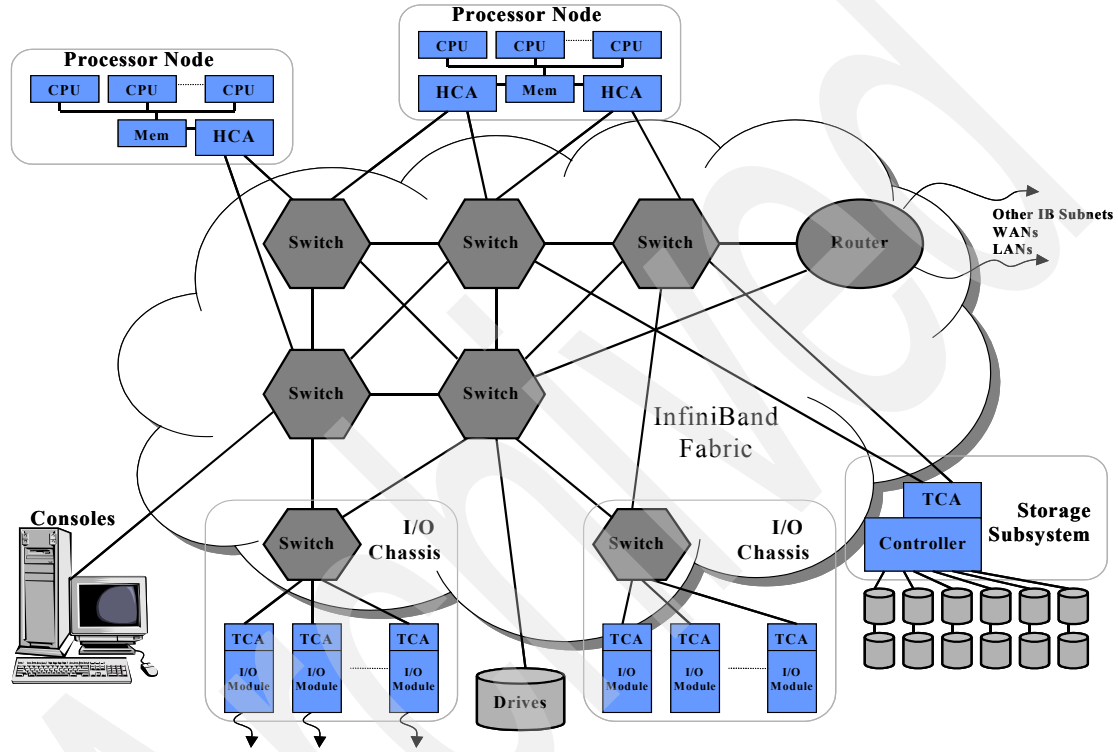


Figure 2-24 InfiniBand System Fabric

While Infiniband is a relatively new interconnecting technology, it has already been adopted by the application scenario High Performance Computing Cluster. The reason is mainly because Infiniband provided superb performance compared to the classic gigabyte Ethernet network.

The advantage of Infiniband technology:

- Increased application performance: high bandwidth, low latency, reduced CPU utilization, and QoS support.
- Support for RAS (reliability, availability, and serviceability).
- A fabric that works both in-the-box and enables Bandwidth Out-of-the-Box.

- Scalability well into the future.

Figure 2-25 shows an IBM PCI-X Dual-port 4x IB HCA adapter.

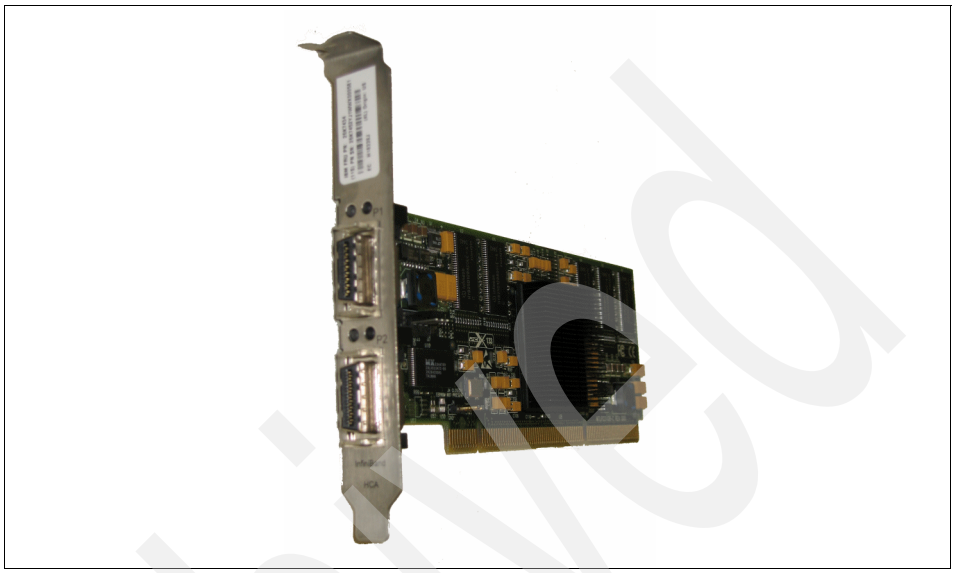


Figure 2-25 IBM PCI-X Dual-port 4x IB HCA adapter

Note: For more information regarding Infiniband, including the technical details of the IBM offering of Infiniband on System p, please refer to the following two Redpapers:

Introduction to InfiniBand for IBM eServer pSeries, REDP-4095

IBM eServer BladeCenter and Topspin InfiniBand Switch Technology, REDP-3949

3

System p server virtualization

This chapter provides a basic overview of the IBM System p server's virtualization capabilities. It provides a brief description of them and provide references to allow a deeper study of this subject.

The following topics are discussed in this chapter:

- Capabilities and concepts
- POWER Hypervisor™
- Partitions in System p servers
- Virtual I/O Server (VIOS)
- Integrated Virtualization Manager and Hardware Management Console
- Considerations about virtualization at the operating system and application level

3.1 System p virtualization capabilities introduction

With the introduction of the POWER5 based System p servers, IBM has introduced significant improvements in the partitioned environment when compared to the previous pSeries server generations.

Just like the POWER4 systems, dedicated logical partitions are still available on System p POWER 5 servers; however, now each logical partition is able to support both dedicated and virtualized server resources while maintaining strict isolation of the assigned resources.

Just like its predecessor, the POWER5 servers use technologies inspired in the System z™ famous partitioning and availability features to allow two types of partitions from the processor point of view:

Dedicated Processor Partitions
Like the POWER4 based servers, the CPUs are exclusively dedicated to the partition and cannot be used by the other partitions under any condition, unless they are properly reallocated to another partition.

Shared Processor partitions or micro-partitions
This is a new type of partition available only on POWER5 based servers. The micro-partition provides the ability to share processors among multiple partitions within the system. This feature allows servers based on the POWER5 processor the ability to map virtual processors to fractions of a physical processor.

Additionally, from the I/O point of view, System p POWER5 servers are able to support more partitions than they have I/O slots due to their ability to share I/O adapters. There are three key types of virtual adapters that allow the virtualized I/O solution to work properly:

Virtual Ethernet Enables a partition to communicate to other partitions without the need of a physical Ethernet adapter.

Shared Ethernet A resource supported by the Virtual I/O Server (VIOS)[1] that provides the partitions with a shared path to an external network.

Virtual SCSI Enables a partition to access block-level storage that is not a physical resource of that partition. With the Virtual SCSI design, the virtual storage is backed by a logical volume on a portion of a disk or an entire physical disk at the VIOS.

[1] We will discuss VIOS in more detail in 3.4, “Virtual I/O Server (VIOS)” on page 60.

With fractional processor allocations and virtualized I/O adapters, more partitions can be created in a given system, which enables clients to maximize the workloads that can be supported on a server and still preserve the isolation of the applications due to the individual operating system images used in each of the partitions. Moreover, the additional workloads supported in the system will allow the client to implement solutions where higher resource utilization can be achieved and deployment of new workloads are considerably simplified.

3.2 The POWER Hypervisor

The technology behind the virtualization of the IBM System p POWER5 servers is provided by a piece of firmware known as the POWER Hypervisor, which resides in flash memory. This firmware performs the initialization and configuration of the POWER5 processor, as well as the virtualization support required to run up to 254 partitions concurrently on the IBM System p POWER5 servers.

The POWER5 Hypervisor supports many advanced functions when compared to the previous version found in POWER4 processor–based systems. This includes sharing of processors, virtual I/O, and high-speed communications among partitions using a virtual LAN, and it enables multiple operating systems to run on the single system. Currently, the AIX 5L, Linux, and i5/OS® operating systems are supported, as shown in Figure 3-1.

With support for dynamic resource movement across multiple environments, clients can move processors, memory, and I/O between partitions on the system as workloads are moved between the partitions.

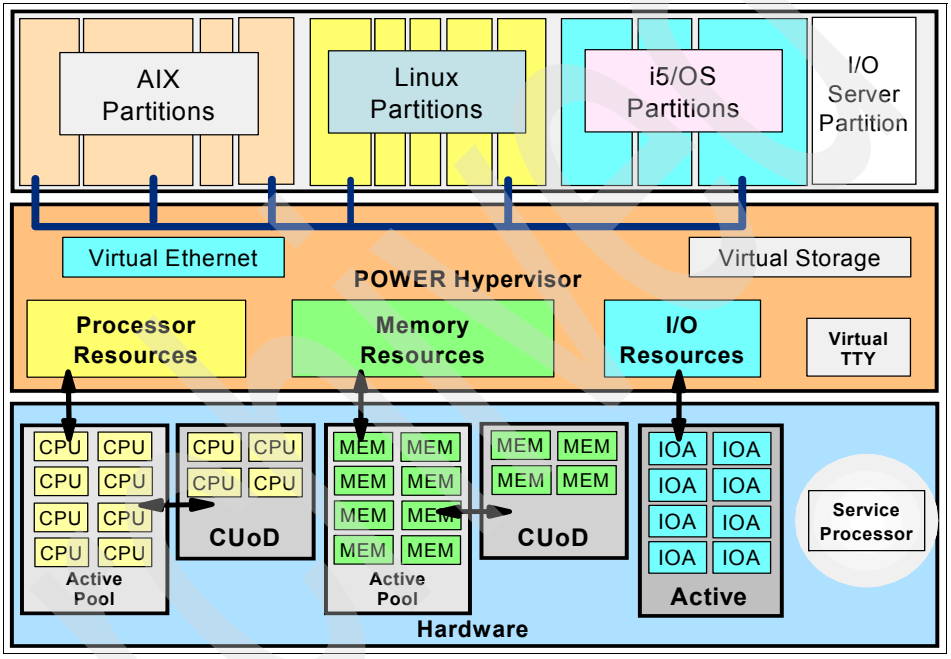


Figure 3-1 Virtualization technologies implemented on POWER5 servers

The POWER Hypervisor is the underlying control mechanism that resides below the operating systems but above the hardware layer (Figure 3-2). It owns all system resources and creates partitions by allocating and sharing them.

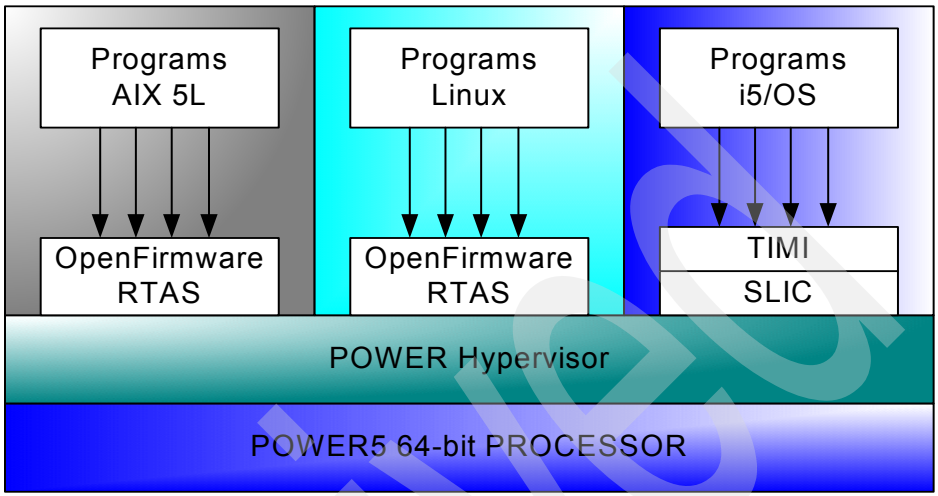


Figure 3-2 System p POWER5 system layers

The layers above the POWER Hypervisor are different for each supported operating system. For the AIX 5L and Linux operating systems, the layers above the POWER Hypervisor are similar, but the contents are characterized by each operating system. The layers of code supporting AIX 5L and Linux consist of system firmware and Run-Time Abstraction Services (RTAS).

System firmware is composed of *low-level firmware* (code) that performs server unique input/output (I/O) configurations and the Open Firmware that contains the boot-time drivers, boot manager, and the device drivers required to initialize the PCI adapters and attached devices. RTAS consists of code that supplies platform-dependent accesses and can be called from the operating system. These calls are passed to the POWER Hypervisor that handles all I/O interrupts.

The distinction between RTAS and Open Firmware is important. Open Firmware and RTAS are both platform-specific firmware and both are tailored by the platform developer to manipulate the specific platform hardware. RTAS encapsulates some of the machine-dependent operations of the IBM System p POWER5 systems into a machine-independent package. The operating system can call RTAS to do things such as start and stop processors in an SMP configuration, display status indicators (such as LEDs), and read/write NVRAM without having to know the intricate details of how the low-level functions are implemented on particular platforms. Open Firmware, on the other hand, does not have not be present when the operating system is running. Open Firmware is defined by the IEEE 1275 standard and is a specification for a machine-independent BIOS that is capable of probing and initializing devices that have IEEE-1275 compliant Forth code in their ROMs. The device tree produced by Open Firmware can then be passed to the operating system when control is passed to the operating system during boot. You can learn more about the IEEE 1275 Open Firmware standard at:

http://www.openfirmware.org

For i5/OS, the Technology Independent Machine Interface (TIMI) and the layers above the POWER Hypervisor are still in place. The System Licensed Internal Code (SLIC), however, is changed and enabled for interfacing with the POWER Hypervisor. The POWER Hypervisor code is based on the System i™ Partition Licensed Internal Code (PLIC) code that is enhanced for use with the System i hardware. The PLIC is now part of the POWER Hypervisor.

Attention: The POWER Hypervisor is mandatory on all POWER5 processor–based systems. This includes any single-LPAR system.

3.3 Logical partitioning on System p

On System p servers, every single resource can be seen as either dedicated or virtualized (also called shared in some documentation). With this simple concept, it is possible to group the logical partitions into three different basic types:

- Dedicated resources partitions
- Micro-partitions with dedicated I/O
- Micro-partitions with shared I/O

One important thing to consider is that although CPU and I/O adapters can be shared, memory is always dedicated and never accessed by a partition that is not the owner of the resource. This is a important feature in order to guarantee partition isolation within a server.

3.3.1 Basic types of logical partitions

Here we discuss the basic types of logical partitions.

Dedicated resources partitions

As the name states, this is a partition where all the resources are always controlled by the specific operating system image. The resources are never accessed by a different partition. Note that idle time in the resources of the partition can never be used by another partition.

Micro-partitions with dedicated I/O

In this type of partition, physical processors are divided into virtual processors that are shared in a pool of one or more logical partitions (LPARs). Note that each LPAR still run its own image of the operating system independently, as in any of the System p partitioning options. Additionally, in this type of LPAR, the I/O resources are exclusively owned by a single partition.

As a benefit of using Micro-partitions, the Hypervisor is able to allocate idle time from an LPAR to another LPAR according to priorities previously set up by the system administrator.

Micro-partitions with shared I/O

Just like the previous LPAR type, the CPU is shared through the virtual processor's resource. Additionally, through the creation of virtual I/O resources, the LPARs are able to share the access to physical I/O resources in order to minimize the total number of adapters as well as less idle time in the physical resources available.

Note: For a complete review on all partitioning technologies currently available within the System p product line, refer to:

Partitioning Implementations for IBM eServer p5 Servers, SG24-7039

3.3.2 Partitions isolation and security

From a functional point of view, applications run inside partitions the same way that they run on a stand-alone System p server. There are no issues when moving an application from a stand-alone server to a partition. The design of a partitioning-capable server is such that one partition is isolated from software that is running in the other partitions, including protection against natural software defects and even deliberate software attempts to break the partition barriers. It has the following security features:

- Protection against inter-partition data access

  The design of partitioning-capable System p servers prevents any data access between partitions, other than using shared networks. This design isolates the partitions against unauthorized access across partition boundaries.

- Unexpected partition crash

  A software failure within a partition should not cause any disruption to the other partitions. Neither an application failure nor an operating system failure inside a partition interferes with the operation of other partitions.

- Denial of service across shared resources

  The design of partitioning-capable System p servers prevents partitions from making extensive use of a shared resource so that other partitions using that resource become starved. This design means that partitions sharing the same peripheral component interconnect (PCI) bridge chips, for example, cannot occupy the bus indefinitely.

With partition isolation and security, you can consolidate applications safely within partitions in partitioning-capable System p servers without compromising overall system security.

3.3.3 Micro-partitions

The POWER5 micro-partition model offers virtualization of system resources. In the model, physical resources are abstracted into virtual resources that are then allocated to partitions. This technology specifies processor capacity in terms of processing units. One processing unit represents 1% of the capacity of one physical core. For example, a partition defined with 220 processing units is equivalent to the power of 2.2 physical cores.

Creating a partition using the micro-partition technology requires a minimum of 10 processing units, or 1/10 of a physical core. Therefore, a maximum of 10 partitions per physical processor can be created; however, for production systems, the practical limit is considerably less. Also, at this point in time, a maximum of 254 partitions is coded into the System p servers.

Another benefit of this technology is to allocate groups of processing units into virtual processors, and present these virtual processors as a processor to the operating system running in the micro-partition. The virtual processors are treated by the operating system just like a real processor, providing better performance for applications that are monolithic and that do not properly release the CPU once they go into wait mode.

Important: A larger number of virtual processors in a micro-partition can prove beneficial for servers running older software. However, most modern applications are designed with a SMP/multi-threaded environment in mind, and if too many virtual processors are defined, it is possible that the context switches that happen to manage the processor virtualization can impact the overall system performance.

Last, but not least, an important feature of micro-partitions is the ability to allocate processor idle time from one partition to other partitions within the server. While setting up the partitions, the administrator is able to set a partition as capped or uncapped. A capped micro-partition is not allowed to receive any additional processor cycles that are idle above its configured capacity. However, an uncapped partition is allowed to consume idle cycles according to a predefined policy. Uncapped partitions can be configured to consume the total idle capacity of the server or up to a percentage of the total idle capacity.

Note: For a complete review on micro-partition currently available within the System p product line, refer to Chapter 4, “Micro-partition technology”, in *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039

3.3.4 I/O virtualization

Just like the micro-partitions, virtual I/O is done through the creation of virtual devices that the operating system recognizes as a real I/O device to allow I/O operations to be executed through a shared physical I/O resource.

The I/O virtualization within System p servers is composed of five different components:

- Three adapters: Virtual SCSI, Virtual Ethernet, and Virtual Serial.
- A mechanism that links the virtual network devices to physical network devices (called Shared Ethernet Adapter).
- A special build of AIX, called Virtual I/O Server (VIOS), installed in a partition to link the virtual adapters to physical adapters.

When combined, these five components allows a partition without any physical adapters to be accessed and access the external physical resources in a client environment.

We will give a more detailed description of these concepts in In 3.4, “Virtual I/O Server (VIOS)” on page 60.

3.3.5 Benefits of partitioning

A partitioned environment provides multiple benefits, especially in an environment based on the POWER5 based system, where the partitioning feature allows for unparalleled granularity for isolated partition. Some benefits we can cite include:

- Consolidating servers

  A logically partitioned server can reduce the number of servers that are needed within an enterprise. You can consolidate several servers into a single logically partitioned system. This eliminates the need for, and expense of, additional equipment.

- Sharing resources

  You can quickly and easily move hardware resources from one logical partition to another as your needs change. Features such as Micro-Partitioning allow for processor resources to be shared automatically among logical partitions that use the shared processor pool. Other features, such as dynamic logical partitioning, allow for resources to be moved to, from, and between running logical partitions manually without shutting down or restarting the logical partitions.

- Maintaining independent servers

  Dedicating a portion of the resources (disk storage unit, processors, memory, and I/O devices) to a partition achieves logical isolation of software. If configured properly, logical partitions also have some hardware fault tolerance. Batch and 5250 online transaction processing (OLTP) workloads, which might not run well together on a single machine, can be isolated and run efficiently in separate partitions.

- Creating a mixed production and test environment

  You can create a combined production and test environment on the same server. The production partition can run your main business applications, and the test partition is used to test software. A failure in a test partition, while not necessarily planned, will not disrupt normal business operations.

- Merging production and test environments

  Partitioning enables separate partitions to be allocated for production and test servers, eliminating the need to purchase additional hardware and software. When testing has been completed, the resources allocated to the test partition can be returned to the production partition or elsewhere as required. As new projects are developed, they can be built and tested on the same hardware on which they will eventually be deployed.

- Running integrated clusters

  Using high-availability application software, your partitioned server can run as an integrated cluster. You can use an integrated cluster to protect your server from most unscheduled failures within a partition.

However, before choosing a partitioned environment, you should consider the following points:

- System p servers are constructed with features to reduce the possibility of system failures. Through the Advanced System Management Interface (ASMI), set the server so that the server can deconfigure failing processors or memory modules automatically. After the server deconfigures the failing processor or memory module, the server continues running without using the deconfigured processor or memory module. With this feature, processor and memory failures are not likely to affect any partitions besides the logical partition(s) to which the specific resource belongs.
- There is plenty of documentation at IBM Web sites that will help a system administrator understand the many concepts and procedures necessary to implement logical partitions successfully on your System p server.

- A consolidated system allows significant reductions in a solution's total cost of ownership (TCO), but resource administration can become a challenging task, particularly in consolidated systems that are used at a level close to their capacity. If you anticipate that you will use your server at a level close to its capacity, consider ordering a server model that is capable of Capacity on Demand (CoD).

3.4 Virtual I/O Server (VIOS)

The IBM Virtual I/O Server is the link between the virtual resources and physical resources. It is a specialized partition that owns the physical I/O resources, and is supported only on POWER5 processor–based servers. This server runs in a special partition that cannot be used for execution of application code.

It mainly provides two functions:

- It serves virtual SCSI devices to client partitions.
- It provides a Shared Ethernet Adapter for VLANs.

The installation of the Virtual I/O Server partition is performed from a special mksysb DVD-ROM that is provided to clients who order the Advanced POWER Virtualization feature. This is a dedicated software only for the Virtual I/O Server operations, so the Virtual I/O Server software is supported only in Virtual I/O Server partitions.

You can install the Virtual I/O Server from DVD using NIM on Linux (NIMoL) from the Hardware Maintenance Console (HMC).

The Virtual I/O Server supports the following operating systems as Virtual I/O clients:

- IBM AIX 5L V5.3
- SUSE Linux Enterprise Server 9 for POWER
- SUSE Linux Enterprise Server 10 for POWER
- Red Hat Enterprise Linux AS 3 for POWER, update 3
- Red Hat Enterprise Linux AS 4 for POWER

The I/O Server operating system is hidden to simplify transitions to further versions. No specific operating system skills are required for administration of the I/O Server.

3.4.1 Virtual adapters

Virtual adapters are constructs presented to the operating system within a partition that are treated just like a physical adapter; however, the adapters are not physically present in the partition.

The POWER Hypervisor provides support for three types of virtual adapters:

- Virtual serial
- Virtual Ethernet
- Virtual Small Computer Systems Interface (SCSI)

The system administrator uses the Hardware Management Console (HMC), Integrated Virtualization Manager, or Virtual Partition Manager to create virtual adapters in order to use virtual I/O devices. Adapters can be added while the system is running. The virtual adapters are recorded in system inventory and management utilities. Converged location codes can be used to correlate operating-system level or partition-level software entities, such as eth0, en0, and CMN21, to adapters. Similarly, Ethernet adapters are visible in the same way as physical Ethernet adapters by the operating system.

Virtual serial adapter

The virtual serial adapter can only be used for providing a virtual console to partitions. This console is accessible to the user in the HMC or in the Integrated Virtualization Manager (IVM) display console.

The virtual serial adapter cannot be used for any other purpose. For example, it cannot be used for HACMP™ heartbeat monitoring.

Virtual Ethernet

Virtual Ethernet enables inter-partition communication without the need for physical network adapters assigned to each partition. Virtual Ethernet enables the administrator to define in-memory, point-to-point connections between partitions. These connections exhibit characteristics similar to physical high-bandwidth Ethernet connections and support multiple protocols (IPv4, IPv6, and ICMP). Virtual Ethernet requires a server based on POWER5 or POWER5+ processors with either AIX 5L V5.3 or the appropriate level of Linux and an HMC to define the Virtual Ethernet devices. Virtual Ethernet does not require the purchase of any additional features or software, such as the Advanced POWER Virtualization feature.

However, in order to bridge the VLAN to a physical LAN, the system requires one of the following:

- Routing through an AIX or Linux partition
- Bridging through Shared Ethernet Adapter through a VIOS partition

Routing can be accomplished by enabling the ipforwarding capabilities of a partition that controls a physical Ethernet adapter. However, this option will use that partition's resources to allow the partition to act as a router.

Bridging can be accomplished by mapping VLANs to one or more physical adapters through the VIOS services. This function acts as a MAC relay (OSI Layer 2) bridge and does not depend on any higher layer protocol. This option requires a VIOS partition; however, virtually no load is added to any of the partitions running an application.

Virtual SCSI adapter

Virtual SCSI is based on a client/server relationship. A Virtual I/O Server partition owns the physical resources, and logical client partitions access the Virtual SCSI resources provided by the Virtual I/O Server partition. The Virtual I/O Server partition has physically attached I/O devices and exports one or more of these devices to other partitions. The client partition is a partition that has a virtual client adapter node defined in its device tree and relies on the Virtual I/O Server partition to provide access to one or more block interface devices. Virtual SCSI requires POWER5 hardware with the Advanced POWER Virtualization feature activated. It provides virtual SCSI support for AIX 5L V5.3 and Linux.

As we write this book, the virtualization features of the POWER5 platform support up to 254 partitions, but the server hardware only provides up to 240 I/O slots per machine. With each partition typically requiring one I/O slot for disk attachment and another one for network attachment, this puts a constraint on the number of partitions. To overcome these physical limitations, I/O resources must be shared. Virtual SCSI provides the means to do this for SCSI storage devices.

Furthermore, Virtual I/O allows attachment of previously unsupported storage solutions. As long as the Virtual I/O Server partition supports the attachment of a storage resource, any client partition can access this storage by using Virtual SCSI adapters.

Note: For further information regarding the architecture and performance considerations for the Virtual I/O Servers, refer to:

Advanced POWER Virtualization on IBM eServer p5 Servers: Architecture and Performance Consideration, SG24-5768

3.4.2 Benefits of Virtual I/O Server

There are multiple benefits that come from a partitioned environment based on virtualized I/O. Most of the benefits are similar to the partitioned environment benefits; however, the virtualized I/O allows for unprecedented levels of resource utilization. For example:

- Consolidating servers

  If I/O is virtualized in a server consolidation solution, the number of I/O cards can be dramatically decreased, which allows a more efficient usage of the bandwidth they provide.

- Sharing resources

  Unlike the partition benefit, the resources are already shared through the virtualization, so more than one logical partition can take advantage of the same resource (I/O adapter) at the same time, therefore removing all the concerns about dynamically allocating the I/O cards to specific partitions as they are required.

- Maintaining independent servers

  Dedicating a portion of the I/O resources to a partition still achieves logical isolation of software due to the virtual adapter architecture design. If configured properly, logical partitions using virtualized I/O are still able to have hardware fault tolerance. Workloads that might not run well together on a single machine can be isolated and run efficiently in separate partitions and still share I/O cards, providing a solution that has better cost benefit ratio.

- Creating a mixed production and test environment

  You can create a combined production and test environment on the same server without the need to dedicate expensive I/O adapters for the test environment, especially in scenarios where test environment utilization may be extremely low.

- Running integrated clusters

  Using high-availability application software, your partitioned server can run as an integrated cluster even under virtualized I/O. You can use an integrated cluster to protect your server from most unscheduled failures within a partition. Unlike most host-based virtualization solutions[2], the System p virtualization solution allows for virtual I/O resources to be linked to physical adapters through multiples VIOS instances within a single server.

[2] The term “host-based virtualization solutions” refers to such solutions as VMWare and HP Integrity VM Manager, which require a host OS to manage all the virtual machines (partitions) created in this environment.

3.5 Integrated Virtualization Manager and Hardware Management Console

To configure and manage logical partitions on a System p server, at least one Hardware Management Console (HMC) is required. For some models of the POWER5 and POWER5+ product line, a HMC can be replaced with the Integrated Virtualization Manager (IVM). Table 3-1 contains a list of all the servers that support IVM to manage logical partitions.

Table 3-1 System p POWER5+ and Blade servers capable of supporting IVM

System p servers	IBM eServer	OpenPower	Blades
p505 p510 p520 p550	p510 p520 p550	710 720	JS21

The HMC is a dedicated computer that provides a graphical user interface for configuring and operating servers that are functioning either in a non-partitioned environment or in a full system partition. More details about the HMC can be found in “Hardware Management Console” on page 27.

On the other hand, the Integrated Virtualization Manager (IVM) was created to allow a more cost efficient and simplified solution management for partitioned environments with virtualized I/O.

3.5.1 Integrated Virtualization Manager basics

The IVM is a browser-based system management interface that is designed to be used to manage a single system without a HMC. The partition creation and management tasks in IVM are simpler than the equivalent functions in a HMC managed solution, especially when comparing the virtual adapter creation and connection.

The VIOS in a managed system uses IVM’s Web-based interface to simplify certain tasks:

- Create logical partitions on a single managed system.
- Manage the virtual storage and virtual Ethernet on the managed system.
- View service information related to the managed system.

However, when a system is managed by IVM in place of an HMC, the system administrator has to consider a few differences while planning the partition scheme:

- All partitions must be micro-partitions. In other words, all CPUs are allocated into the shared processor pool, and then allocated to the partitions as indicated by the partition profile. Entire CPUs still can be dedicated from the pool as in a normal micro-partition.
- All I/O has to be virtualized. At install time, the IVM and VIOS will take control of all physical I/O resources. With IVM, a dedicated I/O adapter is not possible. As a corollary, there is no support for redundant VIOS.
- Dynamic Logical partition operations are supported for memory and CPU only. VIOS 1.3.0.0 or higher is required for this feature.
- Systems plans generated by the system planning tool (SPT) are not able to be deployed to the system.
- Blades JS21 virtualization features can only be managed through IVM. HMC is not an option for this specific offer.

Note: An excellent article about the use of IVM for those interested in deepening their knowledge of IVM can be found at:

`http://www.ibm.com/developerworks/linux/library/l-pow-ivm/`

Additionally, the following Redpaper provides excellent documentation about IVM:

Integrated Virtualization Manager on IBM System p5, REDP-4061

3.5.2 Choosing IVM or HMC to manage your system

IVM was created as a low cost solution that allows fast deployment of simple virtualized System p solutions. It is designed as an easy-to-use tool that allows for point and click configuration of a single entry server running multiple workloads.

If a solution fits in the above description, it is very likely that due to the low cost (especially in the small and medium business sector) that IVM will be the best option.

Meanwhile, if a solution requires a large number of servers, complicated partition environment, capacity upgrade on demand (CuoD), integration with the system planning tool (SPT), and a central point for hardware management or both, a HMC is the best option.

3.6 Considerations at the operating system and application level

Due to the characteristics of the virtualization features in the System p POWER5 servers, the operating system and the application does not realize that they are running in either a micro-partition or a virtualized I/O environment. This allows to all applications to run unmodified in a partition that take advantage of both features.

Additionally, because the VIOS partition will be handling the translation of the virtual adapters I/O operation to the physical adapter, it is extremely important to guarantee that this partition is properly sized to handle the I/O requirements in all partitions. A good source for processor and memory requirements for the VIOS partitions based on I/O requirements can be found at:

http://techsupport.services.ibm.com/server/vios/documentation/perf.html

Another important fact to keep in mind is that because the virtualization is not based on a host operating system running a virtualization software like VMWare or HP Integrity Virtual Machines, it is possible to implement a redundant virtual I/O environment where a partition is able to access a physical adapter in two different VIOS partitions, through the definition of multiples virtual adapters. And once this is done, the operating system can take advantage of high availability features like multi-path I/O software, or link aggregation technologies like Etherchannel, and the whole partition can continue to operate properly even in the case of a fault at the VIOS level or even in the external network or storage devices connected to the server.

In conclusion, with all the capabilities discussed in this chapter, it is easy to understand why the System p POWER5 virtualization capabilities are considered to one of the best products in its target market.

4

Linux

This chapter provides a brief history of Linux, then discusses the reasons for a corporation adopting Linux, and finally demonstrates the advantages of choosing Linux on System p as the platform of choice for Linux deployment.

In this chapter, the following topics are discussed:

4.1 What is Linux

Over the past few years, the Linux operating system has become a real and viable alternative for PC users as well as corporate servers and users. Linux delivers the power and flexibility of a UNIX server or desktop. It also provides a set of utilities, internet applications, and a fully functional desktop interface.

The Linux operating system has become a server platform for powerful internet and many other applications. Linux is capable of running from corporate Web, File Transfer Protocol (FTP), and file and printer servers to wide-area information server (WAIS) Web sites or even corporate database servers and Enterprise Resource Planning (ERP) applications, with real-time clusters or high performance computing clusters.

Linux is a fully functional operating system similar to a UNIX system. It has all the standard features of enterprise UNIX systems. Linux provides an easy way to migrate from other UNIX “flavors” offering the same capabilities and a similar way to work. Management of the command structure is enabled through shells. Enhanced graphical environments and desktops are also available.

Linux is based on four fundamental components:

Kernel This provides low-level system control and interfaces, program and hardware device management, and an abstraction layer for the user level.

init This is the main program executed after the kernel is loaded into memory. From init, you can execute different run levels on the user space, with different characteristics.

File structure This structure is organized on file systems that provide the interfaces and abstraction needed to work with data and files. Files are organized into directories with the disk hardware. Each directory may contain any number of subdirectories, each holding files. This is a similar structure to classical PC and UNIX operating system file structures. Linux supports the majority of the existing file systems, natively or through additional kernel modules.

Applications Linux applications cover many categories. The list is ever-expanding as more companies embrace this technology. Through writing applications for Linux, development costs usually decrease, spawning new projects and increasing the number of titles overall. Across the many horizontal and vertical markets with a Linux presence, there are 11 categories: office productivity, Internet related, network and systems management, software development tools, security, database related, graphic manipulation, audio/video production, data management, accounting and finance, and publishing.

However, Linux has the same multi-user and multitasking capabilities as large UNIX operating systems. It provides the same level of system administration that you find on standard UNIX systems. Users can run several programs concurrently, and there is a separate level for user space and kernel space. You can create user accounts for different users and define their access rights to the system resources: files, hardware components, applications, and others. Installation of new devices and network connection and control is also provided as standard in the Linux operating system.

4.1.1 Historical perspective

Today, nearly everything you read about Linux history begins with the ritual invocation of "... an operating system created in 1991 as a hobby by Linus Torvalds at the University of Helsinki..." While this is true, it does not do justice to the significance of the work and its broad implications for today's users and developers. We know that Linux refers to a UNIX-like operating system. Therefore, we begin with a brief overview of the development of portable operating systems and open source to see the context in which Linux evolved.

The origin of UNIX

In 1969, several AT&T Bell Labs employees began work on an operating system that would come to be called UNIX. A significant novelty of their development was that the operating system was portable across hardware platforms. Prior to this, it was more typical to emulate the (usually) older hardware on a new systems. Such portability was achievable only through writing most of the operating system in a higher level language "above" the hardware.

By 1976, UNIX was being taught in classes on operating systems at the university level, and for various legal reasons permitted free academic access to the source code to UNIX while charging over $20,000 U.S. (in 1976 dollars) for commercial or government access. By the early 1980s, AT&T realized that UNIX

was indeed a source of revenue, and then the publication of the source code in university texts was halted. The era of collaborative programming had arrived.

Berkeley System Distribution, GNU, and free software

In the U.S., the academic variant of interest became the Berkeley Systems Distribution (BSD), where virtual memory and networking were added. These advancements permitted large collaborative projects with contributors being scattered throughout the world. Lawsuits eventually ensued among AT&T, the Regents of the University of California, and other parties over access to and distribution of the operating system source code. Such constraints on intellectual property rights to code, along with commercialization of academic artificial intelligence projects in the late 1970s and early 1980s, provided strong motivation for one researcher from the Artificial Intelligence Laboratories at the Massachusetts Institute of Technology to write an operating system that was portable. This system would also be licensed in a manner to prevent its eventual constraint by intellectual property claims.

The new operating system was to be named GNU, a recursive acronym for “Gnu’s Not UNIX.” It would be licensed under the GNU General Public License (GPL). This was the birth of “free (as in freedom) software”, in contrast to software in the public domain. This is free as in “freedom of speech” not free as in “free food.” To understand the “Four Freedoms” and rules of the GNU GPL, visit the GNU Web site at:

`http://www.gnu.org/philosophy/free-sw.html`

The GNU development effort began by making tools such as editors, compilers, and file utilities in source form that could be compiled and executed on any platform. They standardized and improved upon those offered by commercial vendors.

Around 1990, programmers had contributed a nearly complete operating environment to GNU with the exception of a kernel. The GNU kernel was to be based on a microkernel architecture for improved portability. This approach required an (arguably better) architecture that was completely different from that of a monolithic kernel. The GNU kernel project is known as the *Hurd*.

Linux

With the withdrawal of AT&T source code from the university environment, and the BSD operating system mired in legal challenges by AT&T to its claim to be unencumbered by proprietary code, a small scale UNIX-like skeleton of an operating system called Minix was published in a text to be used as a teaching tool.

It is here that Linus Torvalds enters the story. He decided to write a UNIX-like operating system with improved functionality over that of Minix to run on readily available personal computers. His purpose was to teach himself the C language and improve his understanding of operating system design. He and colleague Lars Wirzenius published their source code under the GPL on the internet for public comment, and Linux was born.

Linux was a kernel without utilities. GNU was an operating environment lacking a finished kernel. And unencumbered non-kernel BSD pieces were available to complete the picture. In short order, the components were combined with installation and maintenance tools and made available by distributors. The first of serious note was Slackware in 1993, which followed by many others, making the GNU/Linux (or simply Linux, as it has come to be known) combination readily available. In only a few years, a worldwide Linux community evolved, comprised of programmers and users attracted by the reliability and flexibility of this “free” operating system.

Since the Linux source code is freely available, several companies have developed different distributions of Linux. A distribution is a complete system. The key component is the Linux kernel. Other utilities, services, and various applications can be included as well, depending on the distribution and the intended use. There is no standard distribution. Each distribution that is available has unique advantages.

Note: For a complete review of Linux’s history, refer to Chapter 1,“Introduction to Linux”, in *Linux Handbook A Guide to IBM Linux Solutions and Resources*, SG24-7000

4.1.2 The Linux distributions

What is usually referred to as Linux is more accurately called a “distribution” or “distro” for short. Technically, Linux is only the core or kernel of the operating system. Linux distributions are prepackaged collections of software, generally comprising the kernel and various other packages.

A Linux distribution includes the Linux kernel plus utilities, programming tools, window managers, and other software that make up a full operating system. Distribution companies such as Red Hat, Novell, Conectiva, and Turbolinux, and non-commercial organizations such as Debian build their own distribution around the basic kernel, which is the core of the distribution. Then they add utilities, such as installation programs and package management tools. Next, they package them on an image that can be presented on a CD-ROM/DVD or downloadable from the internet. The kernel code in each distribution may be different since distributions pull Linux versions. The Linux Standard Base project attempts to

standardize the aspects of a distribution to minimize the changes that may affect portability.

A distributor (such as Red Hat or Novell) is a company who packages a Linux distribution for commercial purposes. They technically do not resell the software license, but rather sell a subscription service for support and maintenance. Generally, these are one-year subscriptions. Distributors also sell Linux consulting and integration services. They may sell applications or middleware in addition to the Linux operating system.

There are literally hundreds of distributions documented on the internet. A great resource to keep track of existing distributions is the DistroWatch portal that is updated daily:

http://distrowatch.com/

Due to the dynamic nature of Linux, the top ten ranked distributions change constantly. If we do not mention your favorite distribution, it is not an oversight. Rather, we highlight those that are currently most visible within the scope of System p servers. The Linux world is in constant change, so we encourage you to investigate the resources yourself to find the latest information.

Although many distributions are available for System p servers, the IBM software stack has been tested and provided a stable environment to handle enterprise class workloads running with distributions listed at:

http://www.ibm.com/systems/p/linux/

4.1.3 The Linux Standard Base project

The Linux Standard Base project is an attempt designed to bring the array of Linux distributions into a common core and avoid the incompatibility issues that plagued the different UNIX flavors.

As a workgroup of the Free Standards Group, it is supported by the development community and IT industry leaders, and it is an independent, non-profit organization dedicated to accelerating the use and acceptance of open source technologies through the development, application, and promotion of standards. These standards include common behavioral specifications, tools, and APIs to make development easier across Linux distributions.

The majority of Linux distributions that are available on the market tend to meet the Linux Standard Base definitions and requirements to make a unique common base for all distributions.

For more information, see the Linux Standard Base home page:

http://www.linuxbase.org

4.2 Why Linux

The majority of enterprise Linux implementations are found in corporate intranets or in the internet infrastructure. Other notable uses include file and print server deployment, application development, messaging server deployment, data warehousing, embedded applications, thin client applications, and massively parallel scientific applications. Linux is becoming robust enough to find its way into engineering software development, the health care industry, the retail industry, and financial institutions.

Wall Street has adopted Linux in a big way. Major companies have been in the press, sporting their new implementation projects with Linux. These are not just departmental solutions, but large company-wide solutions indicating potential savings of 30 to 1. These results will certainly attract the attention of Chief Financial Officers (CFOs) and Chief Information Officers (CIOs) looking to improve their business with leading edge solutions and major spending reductions.

Linux is also taking hold in the public and government sectors as the IT industry tries to reduce expenditures and create a flexible and transparent environment. How does it compare to the traditional UNIX systems, vendors, or Windows®? Does Linux provide a real advantage?

The following topics will cover the benefits that Linux can bring to an enterprise, both from the technical as well as the financial point of view.

4.2.1 The open source software advantage

Open source refers to any program whose source code is made available for use or modification as users or other developers see necessary.

Few people know that open source was the original business model for software development! In the 1960s, nobody would buy a computer (a huge investment at that time) that was not immediately ready for use. Software had to be given away by manufacturers as “a way to sell the hardware faster,” and free of charge for that reason. The source code was distributed so that anybody could change it. At the time, nobody would or could use a computer without having programming skills.

But open source software (OSS) is different in nature. It is usually developed as a public collaboration and made *freely available in a machine-independent format*. Most of the software is written in C. One of the first tools written by Richard M. Stallman (RMS), founder of the Free Software Foundation, was an Open C compiler called the *gcc*.

Most software, especially commercial software, is shipped only in a binary format and its source code is heavily protected from publication. This preserves the methods and technologies of the software creator, who typically invests a significant amount of money in their development.

However, OSS software not only “ships” with the binary format, but it also provides the source code for the users and developers to make changes and understand how the software operates.

The availability of the source code is where a key advantage of the OSS model lies. With OSS based solutions, technical IT staffs can develop a better understanding of the software in their environment, adapt the software for their needs, and submit the change back to the development community to improve the code through extensive testing and tuning (some OSS projects have over 20,000 contributors).

Also, innovation within the open source community typically occurs at a higher rate and volume than in closed-source communities due to the nature of OSS, where larger number of developers are involved in the project.

Additionally, the source code availability allows for a platform independent environment, where binary files can be generated for any platform. This last “feature” of OSS allows enterprises to adopt a solution that is not locked to a specific hardware or software environment, and therefore empowers the IT staff within a corporation to make better choices.

Consideration: Because OSS exists with multiples software licenses, a developer has to be very careful with the rights and obligations related to derivative work done with any OSS code.

A very good discussion of the license models used by the OSS community can be found at Chapter 2,“Open source software”, in *Linux Handbook A Guide to IBM Linux Solutions and Resources*, SG24-7000.

4.2.2 Return of Investment

Return of Investment (ROI) is equally significant in the client decision-making process. CFOs take a good look at the ROI they can expect to realize with new servers. While it may be useful to consider the overall total cost of ownership, clients are interested in saving money in a tough economic environment.

ROI can be more effective than Total Cost of Ownership (TCO) as a measure of business value, especially with new, emerging technologies. For this reason, we expect to see a movement of the business community toward a more comprehensive ROI justification, where TCO is just a portion of the overall analysis.

Many companies are beginning to view ROI analysis as a higher quality approach for making business decisions since it is more inclusive of the risks and returns of making new IT investments. This is what CFOs are looking for, and why Linux is becoming more accepted each day.

Numerous case studies have shown Linux to deliver outstanding ROI value. These benefits may derive from much greater deployment flexibility, much greater openness, much higher performance and functional benefits (derived from the prevalent deployment architectures described earlier), and universally accepted better reliability and security. Such emerging technologies as Linux typically excel in ROI analyses, which highlight the benefits of the new technology, one of which is cost savings.

4.2.3 IBM and open source

IBM is committed to open source as both a license and a development model for several reasons:

- IBM clients and partners have requested open source software (including Linux) support for all IBM platforms, products, and solutions.
- Open source software, with its wide distribution and use, typically becomes an industry standard.
- Innovation within the open source community typically occurs at a higher rate and volume than in closed-source communities.

In fact, the first IBM open source license was the IBM Public License (IPL), introduced in 1999 and certified by the OSI in the same year. As part of the Eclipse release in 2001, IBM updated the IPL and renamed it the Common Public License (CPL). The OSI certified the CPL in May 2001 as an open source license.

For more information about the CPL, refer to:

http://www.ibm.com/developerworks/opensource/library/os-cplfaq.html

Additionally, IBM is a major contributor to the open source community, with over 250 developers worldwide working full time on several initiatives. Such initiatives include:

- Linux Technology Center (LTC): The LTC works to enable the enterprise capabilities of Linux through development and contribution of technology, utilities, tools, and code. You can learn more about the LTC on the Web at:

  http://www.ibm.com/developerworks/linux/ltc/

- Open source software support: IBM is an active supporter of projects and software including:
  - Open Source Cluster Application Resources (OSCAR)
  - Open Source Developer Lab (OSDL)
  - GNOME
  - KDE
  - Open Source Initiative (OSI)
  - Free Standards Group
  - USENIX
  - Linux high availability
  - OpenLDAP
  - USB
  - PCI hot plug
  - Advanced Power Management™ (APM)
  - PPC-32 and PPC-64
  - Stream Control Transmission Protocol
  - Free Standards Group (Linux Standard Base)
  - Samba
- Eclipse: This is an open source software integrated development environment (IDE) sponsored by IBM that acts as a development infrastructure platform through extensions (called plug-ins) to the base IDE. Eclipse can provide nearly any type of capability, including code version management functionality and even syntax highlighting. IBM WebSphere Studio Workbench is a commercial product that is built on Eclipse.

You can find Eclipse on the Web at:

http://www.eclipse.org

4.3 Why Linux on System p

Linux on System p is a key part of the IBM Linux strategy. During the past few years, IBM has been developing the ability to integrate Linux capabilities into the System p servers. System p servers can run Linux alone, or can run both IBM AIX and Linux in logical partitions (LPAR) of the same server.

IBM as a corporation has embraced Linux as though it were one of its own internally developed operating systems. Linux receives full support from all IBM software development organizations, including Lotus®, Tivoli, DB2, and WebSphere, and many of these software products are available for Linux. The IBM Server division has enabled Linux on all IBM server platforms from the x86-based System x™ server to the PowerPC-based System i and System p servers, to the System z mainframe. IBM has a Linux Technology Center staffed with more than 750 IBM employees worldwide working full time on various open source projects, including Linux. The IBM Services division offers training, consulting, and support to its clients.

This section provides information about Linux enablement for the System p server. It discusses the ways that you can integrate Linux applications into commercial System p environments and various advantages of selecting the System p server as the Linux deployment platform of choice.

4.3.1 How Linux runs on System p servers

The System p server is one of the IBM key solutions to fit a variety of client requirements. System p servers offer leading edge performance, and can help you confidently manage your business' growth and risks.

To meet client requirements, System p products offer a full range of high-performance servers, from a single CPU system all the way to 64-core SMP systems, complete with highly functional, state-of-the-art hardware and software.

As of today, all the System p product line is able to run Linux natively in a full partition system, or within a logical partition (LPAR). When running within a partition, the Linux instance can take advantage of all virtualization features mentioned in Chapter 3, "System p server virtualization" on page 49. Moreover, AIX is not required in the machine in order to install Linux, partition the system, or both, although they can coexist in different partitions at the same time within a single system.

Linux applications can run on a System p LPAR or in a full partition mode hardware by simply installing the Linux operating system and then installing the required applications. However, you need to obtain the binaries for the PowerPC versions of Linux in order to run the applications on a System p server.

If such binaries are not available, then downloading the application source code,

Tip: IBM has recently announced a statement of direction where System p POWER5, POWER5+, and PowerPC based blades servers will be able to install and run x86 Linux binaries.

More details about this statement of direction can be found in the IBM United States Hardware Announcement Letter 106-627 from August 15, 2006.

assuming they are available, and simply compiling it on a System p server will allow the generation of the proper binaries files. These files will be ready to be executed natively on the System p Linux environment.

4.3.2 Linux on System p scalability

Unlike an Intel-based system, large SMP systems tend to be considered 8-core (system with 8 cores) or sometimes even smaller[1]. On the other hand, System p servers are able to grow up to 64-core and still provide near linear scalability. A quick review of the p595 tpm-c numbers that can be found at:

http://www.tpc.org

is an excellent proof of this statement.

This System p server capability provides two major benefits for clients:

- Much larger range of growth capability for applications that scale well vertically
- Much larger server consolidation foot print that allows for better overall resource utilization

However, this section will concentrate on the first benefit, vertical scalability, and how key it is for System p large servers. However, keep in mind that Linux on System p offers a choice of scale-out, scale-up, and virtualized scale-out to support a wide range of workloads, giving clients unparalleled flexibility in their

[1] Although Intel-based systems with 32 CPUs exist, in real world applications, they tend to be partitioned in order to guarantee acceptable performance levels at the server, as serious bandwidth issues exist when compared to 16-core systems in the System p product line.

solution design, especially when the scope of products available in the System p product line is taken into consideration.

The key component to make Linux on System p servers a very scalable solution was the improvements made in the 2.6 Kernel. These improvements allow for effective scalability over 8-core systems/partitions and memory addressing of much larger memory quantities when compared to its predecessor.

There is an excellent publication, *Towards Linux 2.6: A look into the workings of the next new kernel* by Santhanam. This publication is available on the developerWorks Web site at:

http://www.ibm.com/developerworks/linux/library/l-inside.html

It discusses in detail the scalability of the Linux 2.6 kernel, and how this version of the kernel differs in scalability from the 2.4 kernel. With regard to scalability on SMP systems, the article describes how the scheduler works in the 2.4 kernel and the improvements made in the 2.6 kernel. The 2.6 kernel has redesigned SMP scheduling algorithms and is expected to perform well in 8-core to 64-core or even larger SMP environments. Some of these 2.6 kernel features have been back-ported to the 2.4 kernel. This article contains many links that probe more deeply into detailed areas relating to the main article.

Additionally, IBM did some extensive testing of the kernel for the 2.6 release. The following link shows a example of what was done:

http://www.ibm.com/developerworks/linux/library/l-web26/

Although the tests were done in a 8-core Pentium® III machine in a static Web page benchmarking test, the results are an excellent way to show key the 2.6 kernel is in order to provide better scalability to larger SMP system.

To summarize the test, by keeping all the software the same and only upgrading to a 2.6 series kernel from a 2.4 series kernel, a 600% increase in performance occurred, as demonstrated by Figure 4-1 on page 80.

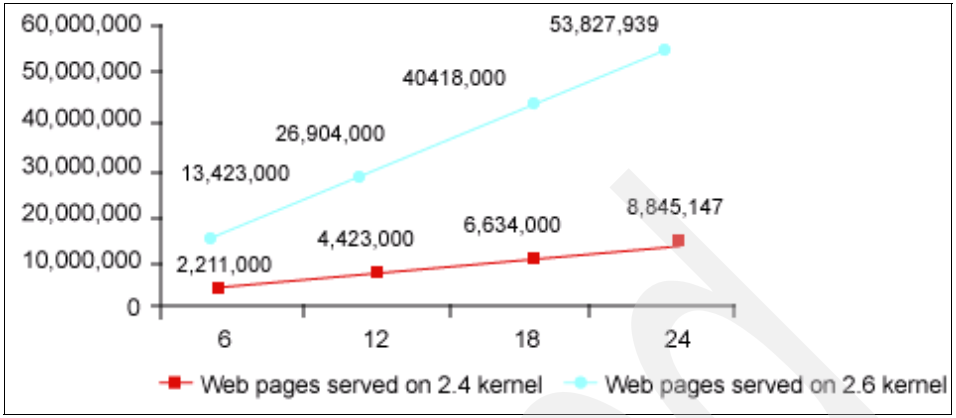


Figure 4-1 Web pages served versus time in an 8-core system

4.3.3 Linux for System p and Reliability, Availability, and Serviceability

A key attribute of the System p servers is its critical Reliability, Availability, and Serviceability (RAS) features. RAS is a concept used not only by IBM, but by virtually all the vendors when designing their hardware and software platforms.

However, the implementation of RAS features and functionality may vary depending on the platform and environment on which you are working. Linux for System p is rapidly developing RAS capabilities that are not matched by many of the other Linux platforms available. As of the writing of this book, some of the advanced System p RAS features are only fully realized when running AIX.

More information about RAS features on System p servers running Linux can be found in the white paper *Reliability, Availability, Serviceability (RAS) for Linux on POWER* by Sze, which can be found at the following URL:

`http://www-1.ibm.com/partnerworld/pwhome.nsf/weblook/pat_linux_learn_tech.html`

The following pSeries RAS features, offers, and tools are supported when running the most current Linux distribution builds[2] on a System p server:

- System wide Features:
  - Automatic First Failure Data Capture and diagnostic fault isolation capabilities.
  - Fault tolerance with N+1 redundancy of power and cooling, dual line cords, and concurrent maintenance for power and cooling.

[2] Some older versions of the distribution may not support all the features listed in this section.

  - Predictive failure analysis on processors, caches, and memory.
  - Fault avoidance through highly reliable component selection, component minimization, and error mitigating technology internal to chips.
  - Concurrent runtime diagnostics for power and cooling.
  - Service processor (with redundancy option available in some System p models) and Service Agent.
  - Service Processor failover at system IPL time.
  - Service Focal Point on the Hardware Management Console (HMC).
  - High Availability Cluster Multiprocessing (HACMP). Additional third party and Open Source solutions are also available.
  - Dynamic logical partitioning (memory DLPAR operations are still not supported).
- Processor Features
  - Self-healing internal POWER5 and POWER5+ processor array redundancy.
  - Error logging and reporting.
  - Boot-time processor deallocation (Persistent Processor Deallocation).
  - Dynamic Processor Deallocation.
  - Dynamic Processor Sparing at runtime (either through CuOD processors or spare capacity from an active share pool).
- Memory Features
  - Scrubbing and redundant bit-steering for self-healing in main storage.
  - Chipkill and ECC correction in main storage with dynamic bit-steering and memory scrubbing.
  - ECC in memory, L2 and L3 cache, and L1 parity check plus retry.
  - Boot-time memory deallocation.
  - Memory error correction extensions.
- Storage System
  - Journaled file system (several available under Linux).
  - Service processor (with a redundancy option available in some System p models) and Service Agent.
  - Hot swap disks (internal and external).
- I/O Systems (PCI and RIO buses)
  - PCI Extended Error detection

- PCI bus parity error recovery
- PCI card Hot Swap
- Slot freeze detection
- Unrecoverable error handling
- Redundant RIO link

Some of the pSeries RAS features that are not supported in Linux as of the writing of this book are:

- Dynamic Firmware update with HMC
- Limited runtime diagnostics (some exist, but not at the same level as AIX)
- PCI-PCI Bridge extended error handling
- PCI Extended error correction is limited
- GX+ Bus persistent deallocation
- Manual Memory sparing from active pool

4.3.4 Virtualization and partitioning capabilities

The IBM System p Architecture servers support a very flexible and powerful virtualization and partitioning environment. The basics of these capabilities were described in Chapter 3, “System p server virtualization” on page 49.

Clients all over the world are taking advantage of these advanced features to take the returns from their System p server investment to a whole new level, with reported cases where the infrastructure cost was reduced by up to 62%. Information about this study on cost reduction and other client quotes and how they benefit from the System p virtualization and partitioning capabilities can be found at:

http://www-03.ibm.com/systems/p/apv/

In fact, clients are already realizing cost and utilization gains when running Linux on their System p servers. The Bionorica case study available through the link above can show details on such matters. This is possible because all these advanced virtualization and partitioning features[3] are available under Linux when running on System p servers.

Having the ability in a single server to create virtual servers to support workloads with security and integrity provides a low cost solution to gain efficiencies not available to other 64-bit architectures, specially in the Small and Medium Business (SMB) marketplace.

4.3.5 Industry proven architecture

The IBM POWER Architecture, as found in System p servers running Linux, is based on architecture with over 20 years in the market. Moreover, the Hypervisor, a hardware and firmware based solution with a System z server heritage of over 40 years of excellency and innovation, provides partitioning

[3] There is one partitioning/virtualization feature not available under Linux on System p, that is, a DLPAR operation adding or removing memory. The Linux kernel at this point in time does not have the necessary capabilities to allow such operations.

capabilities that allow the optimization of system resources with extremely low load within a proven and tested architecture design that has a background history that few corporations are able to claim for their own solutions.

The POWER Architecture is highly acclaimed and accepted in the industry, supporting a wide range of applications ranging from embedded systems to consumer devices to large supercomputers. The POWER Architecture is pervasive in the world of information technology (IT) today. And the latest member of this family, the POWER5+, has a very strong road map to provide peace of mind when choosing a strategic architecture for 64-bit Linux solutions.

4.3.6 End-to-end solutions and support

With Linux for System p, you can incorporate the best of open source applications into your environment. Linux for System p makes it possible to enrich the capability of any pre-existing System p AIX environment with the power and benefits of open source software. IBM and other third-party companies are currently working on providing applications for Linux on System p. For a list of some of the current developments in application solutions for Linux on System p, refer to:

http://www-03.ibm.com/systems/linux/power/apps/all.html

This Web page lists more than 500 applications available to run in Linux on System p, with most being linked through an URL link to the particular product's Web page. There is also a convenient filter that lets you list application offerings by category.

These applications range from batch job workload management to compilers, and include such industries as biomolecular research for cancer.

Some important commercial applications in the list above are:

- Database
  - Oracle
  - DB2
  - Informix®
  - Sybase
- ERP
  - SAP
  - Oracle
- Web Solutions
  - WebSphere
  - Weblogic
  - Complete OSS solution stack (Apache, PHP, Tomcat, JDK™, and JRE™)

In conclusion, with all the support and capabilities of running Linux in a System p environment, it is easy to understand why the System p servers are an excellent choice to run your Linux solutions.

5

Linux monitoring tools

The Linux operating system supports various types of applications, such as DB2, Web Sphere, Oracle, Apache, and so on. Operating system performance can vary based on the running applications and workloads. For applications to run efficiently, the performance of the operating system must be tuned to suit the specific applications and workload. Regular monitoring is required to identify performance bottlenecks and tune the system per the workloads. Linux provides several tools to monitor the hardware and software components. This chapter describes the monitoring tools available for POWER platforms that will help you identify and analyze the performance bottlenecks and tune the system accordingly. The tools mentioned are available on SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux As-4 update 4.This chapter also introduces the new tracing and universal monitoring, also known as *SystemTap*.

Based on the usage, the following monitoring tools are described:

5.1 Universal system monitoring tool: SystemTap

As Linux usage in the enterprise world is growing day-by-day, the performance analysis is increasingly performed by the system analysts rather than the system developers. System analysts are exposed to more complex performance problems, where the existing tools like **vmstat**, **mpstat**, **iostat**, and **top** do not provide sufficient information to identify performance problems. SystemTap was developed to dynamically instrument the kernel and provide enough data to the system analysts to analyze the performance problems. SystemTap instrumentation incurs low load when enabled and zero load when disabled. SystemTap provides facilities to define instrumentation points in a high level language and to aggregate and analyze the instrumentation data similar to Suns Dtrace tool. SystemTap provides a simple command-line interface and scripting language for writing kernel instrumentation. SystemTap is an end-to-end tool that can be used to monitor the system and the complete system performance data. SystemTap can be used together with information from a specific kernel routine, which will help identify the exact cause for the bottleneck. SystemTap has been a universal system monitoring tool because none of the existing tools in Linux provide the information from the sub system level to routine level granularity.

The usage of SystemTap is discussed below.

Tip: To use SystemTap, you need to install kernel-debuginfo-2.6.9.42.EL.ppc64.rpm for Red Hat Enterprise Linux AS 4 Update 4.

stap

The **stap** program is the front end to the SystemTap tool. It accepts probing instructions (written in a simple scripting language) and performs the requested system trace/probe functions. A SystemTap script can be as simple as a single line, as shown below:

```
# stap -e 'probe syscall.open {printf("%s calls open(%s)\n",
execname(), argstr)}'
```

You can also download an example script from:

http://sourceware.org/systemtap/wiki/WarStories

Tip: You can get more information about Systemtap from:

http://sourceware.org/systemtap/wiki

5.1.1 Getting started

You can trace any function in the kernel from the command line. Let us start with a simple example to trace the entry of the function sys_mkdir; when the user tries to create any directory, the following command logs it:

```
#stap -e ‘probe kernel.function (“sys_mkdir”) { log ("dir created") }’
```

Note: To stop the **stap** script, press Ctrl-C.

The output of the **stap** command to trace the entry of function sys_mkdir() is shown in the Example 5-1, where the user creates two directories, hence “`dir created`” appears twice.

Example 5-1 Shows the output of stap to the trace entry of the function sys_mkdir ()

```
#stap -e ‘probe kernel.function (“sys_mkdir”) {log(“dir created”)}’
#dir created
#dir created
#Ctrl+C
```

You may want to print all the callers to open() syscall with the function arguments. The output of the **stap** command shown in the Example 5-2 traces sys_open(), while the **gam_server** command is executing the server open syscall to read “/root/Templates”. The **stap** command inputs the probe on kernel routine sys_open():

```
#stap -e ‘probe syscall.open {printf(“%s: %s\n”, execname(), argstr)}’
```

Example 5-2 Shows the output of stap to print callers to sys_open()

```
# stap -e 'probe syscall.open {printf("%s:
%s\n", execname(), argstr)}'
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
```

```
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
gam_server: "/root/Templates",
O_RDONLY|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK
```

Tip: To list all the kernel routines that can be probed, use the following command:

```
stap -p2 -e 'probe kernel.function("*") {}' | sort | uniq
```

Now, you may want to print all the programs that make calls to read() syscall with the function arguments. The output of the **stap** command shown in the Example 5-3 traces sys_read(), where the smbd is issuing server read syscalls:

```
#stap -e ‘probe syscall.read {printf(“%s: %s\n”, execname(), argstr)}’
```

Example 5-3 Shows the output of stap to print callers to sys_read()

```
smbd: 23, [0x0000000f7d0a008], 4
smbd: 23, [0x0000000f7d0a008], 4
smbd: 23, [0x0000000f7d0a00c], 4160
smbd: 23, [0x0000000f7d0a00c], 4160
smbd: 23, [0x0000000f7d0a008], 4
smbd: 23, [0x0000000f7d0a00c], 50
smbd: 23, [0x0000000f7d0a008], 4
smbd: 23, [0x0000000f7d0a00c], 4160
smbd: 23, [0x0000000f7d0a008], 4
smbd: 23, [0x0000000f7d0a00c], 4160
smbd: 23, [0x0000000f7d0a008], 4
rhn-applet-gui: 14, [0x0000000ffffced8], 32
smbd: 23, [0x0000000f7d0a00c], 4160
smbd: 23, [0x0000000f7d0a008], 4
smbd: 23, [0x0000000f7d0a00c], 4160
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
stpd: 3, [0x000000010016ed8], 8192
```

5.1.2 Demo examples

When tracing various kernel components or subsystems, the script size often increases and it becomes difficult to write the script on the command line. In order to facilitate system analysis to write and use more complex scripts, SystemTap users can write a script and run it through **stap**.

You might want to know the number of system calls issued by the various applications. A script to trace all the system calls and report the top 20 syscalls can be downloaded from the following URL:

http://sources.redhat.com/cgi-bin/cvsweb.cgi/src/examples/small_demos/top.stp?rev=1.3&content-type=text/x-cvsweb-markup&cvsroot=systemtap

The output of the **stap** command lists the top 20 syscalls executed on the server running I/O intensive jobs and it updates the output every five seconds, as shown in the Example 5-4.

Example 5-4 Showing the output of stap to list the top 20 syscalls in five seconds

```
# ./top.stp
SYSCALL                          COUNT
sys_kill                         1170229
sys_read                          2032
sys_fcntl                         1723
sys_poll                           715
sys_ioctl                          631
sys_send                           529
sys_sendto                         529
sys_write                          432
sys_pwrite64                       409
sys_stat64                         109
sys_munmap                         106
sys_mmap                           106
sys_lstat64                         43
sys_newstat                         25
sys_gettimeofday                    14
sys_close                           13
sys_newlstat                        10
sys_rt_sigprocmask                   9
sys_fstat64                          6
sys_open                             5
sys_rt_sigaction                     2
-------------------------------------
```

You may want to trace the syscalls on per process basis or even for a particular command. Here is a sample script that traces the syscalls on a per process basis or per command basis; it can be downloaded from the following URL:

http://sources.redhat.com/cgi-bin/cvsweb.cgi/src/examples/small_demos/prof.stp?rev=1.1&content-type=text/x-cvsweb-markup&cvsroot=systemtap

The output to list the syscalls and time spent in the syscall per-process basis is shown in the Example 5-5. This example lists the system calls for process ID 2424.

Example 5-5 The output of the stap command to list syscalls per-process basis

```
#./prof.stp -x 2424
sys_mmap                calls:   125    avg time (ms):    7
total(ms):    978
sys_munmap              calls:   126    avg time (ms):  574
total(ms):  72449
sys_read                calls:   769    avg time (ms):    8
total(ms):   6170
sys_lstat64             calls:    72    avg time (ms):    5
total(ms):    430
sys_fstat64             calls:     3    avg time (ms):    4
total(ms):     12
sys_fcntl               calls:   825    avg time (ms):17855
total(ms):14730742
sys_stat64              calls:   173    avg time (ms):    9
total(ms):   1594
sys_sendto              calls:   379    avg time (ms):   18
total(ms):   7123
sys_kill                calls:1763632   avg time (ms):    2
total(ms):3691247
sys_pwrite64            calls:   202    avg time (ms):   40
total(ms):   8114
sys_ftruncate           calls:     2    avg time (ms): 1334
total(ms):   2669
sys_close               calls:     2    avg time (ms):    7
total(ms):     14
```

Tip: To profile syscalls issued by a specific user command, run:

```
#prof.stp -c "top -n1"
```

This will start up "top" and after one iteration, it will exit. Here we are trying to profile top command. You profile any of such commands using prof.stp.

You would like to monitor the time taken for process specific events, such as process creation, process start, process exec, process exec completion, and process release. The SystemTap script prof_snoop.stp lists the process specific event, and can be downloaded from:

http://sources.redhat.com/cgi-bin/cvsweb.cgi/src/examples/small_demos/proc_snoop.stp?rev=1.4&content-type=text/x-cvsweb-markup&cvsroot=systemtap

The output of the **stap** command to trace process events is shown in Example 5-6, where each command has create, start, exec, and exit events. The time taken to complete each command is the difference between the end time and the start time of each command. The user issued the **uname** and **ls** commands, where **uname** took 1431 microseconds to complete and **ls** took 4831 microseconds.

Example 5-6 Shows the output of the stap command to trace a process specific event

```
# stap proc_snoop.stp
   TIMESTAMP    PID    TID  EXECNAME          ACTION
     1001685   5050   5050  bash              create p:6618 t:6618
n:bash
     1001700   6618   6618  bash              start
     1002100   6618   6618  bash              exec /bin/uname
     1002443   6618   6618  uname             exec success
     1003116   6618   6618  uname             exit 0
     1003336   5050   5050  bash              remove p:6618 t:6618
n:uname
     7216399   5050   5050  bash              create p:6619 t:6619
n:bash
     7216411   6619   6619  bash              start
     7216875   6619   6619  bash              exec /bin/ls
     7217215   6619   6619  ls                exec success
     7221230   6619   6619  ls                exit 0
     7221609   5050   5050  bash              remove p:6619 t:6619 n:ls
     8022654   5050   5050  bash              create p:6620 t:6620
n:bash
     8022668   6620   6620  bash              start
     8023145   6620   6620  bash              exec /usr/bin/clear
     8023512   6620   6620  clear             exec success
     8024519   6620   6620  clear             exit 0
```

SystemTap can be used together with information about the kernel memory allocations. The sample script to trace all the kernel memory allocations can be downloaded from:

http://sources.redhat.com/cgi-bin/cvsweb.cgi/src/examples/small_demos/kmalloc.stp?rev=1.1&content-type=text/x-cvsweb-markup&cvsroot=systemtap

The output of the **stap** command to trace the kernel memory allocations is shown in Example 5-7, where most of the kernel memory allocation requests were for 2048, 512, and 32 bytes.

Example 5-7 Output of stap for kernel-memory allocation statistics

```
#./kmalloc.stp
Count:   95398 allocations
Sum:     140420 Kbytes
Average: 1471 bytes
Min:     13 bytes
Max:     4096 bytes

Allocations by size in bytes
value |-------------------------------------------------- count
    2 |                                                       0
    4 |                                                       0
    8 |                                                       2
   16 |                                                       5
   32 |@@@@@@@@@@@@@@                                     15232
   64 |@                                                   1420
  128 |                                                     858
  256 |                                                     442
  512 |@@@@@@@@@@@@@@@@@@@@                                22400
 1024 |                                                       0
 2048 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  55034
 4096 |                                                       5
 8192 |                                                       0
16384 |                                                       0
```

SystemTap can also trace kernel memory allocations specific to a particular process or set of processes. The script to trace the kernel memory allocation specific to a particular process can be downloaded from:

http://sources.redhat.com/cgi-bin/cvsweb.cgi/src/examples/small_demos/kmalloc2.stp?rev=1.1&content-type=text/x-cvsweb-markup&cvsroot=systemtap

The output of the **stap** command to trace the kernel memory allocation per-process wise is shown in Example 5-8, where the memory is allocated by the swapper and gnome-terminal processes. Swapper allocates more memory chunks of size 512 and 2048 bytes and the gnome-terminal allocates memory chunks of 64 bytes.

Example 5-8 Showing the output to trace kernel memory allocation per-process wise

```
# ./kmalloc2.stp
Allocations for swapper
Count:   21784 allocations
Sum:     38643 Kbytes
Average: 1773 bytes
Min:     568 bytes
Max:     2360 bytes

Allocations by size in bytes
value |-------------------------------------------------- count
  128 |                                                      0
  256 |                                                      0
  512 |@@@@@@@@@@@@@@@@@@@@@@@@@@                         7539
 1024 |                                                      0
 2048 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  14245
 4096 |                                                      0
 8192 |                                                      0

----------------------------------------------------

Allocations for gnome-terminal
Count:   21 allocations
Sum:     2 Kbytes
Average: 104 bytes
Min:     104 bytes
Max:     104 bytes

Allocations by size in bytes
value |-------------------------------------------------- count
   16 |                                                     0
   32 |                                                     0
   64 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    21
  128 |                                                     0
  256 |                                                     0

----------------------------------------------------
```

5.1.3 SystemTap Tapset

A SystemTap tapset is a collection of probepoints and the code to export variables/data from the probed routine for use in Systemtap scripts. A tapset can be written in scripting language. For example, system call tapset is a collection of probe points for all the system calls, and it also exports such interesting date as the name, function arguments, trace string, and so on. A networking tapset to trace the main device receive and transmit routines, shown in Example 5-9, exports such data as the network device, actual length of the data, real size of the buffer, and the protocol. This tapset is located in the /usr/share/systemtap/tapset directory.

Example 5-9 Shows systemtap networking tapset from /usr/share/systemtap/tapset/

```
%{
#include <linux/netdevice.h>
%}

/* Main device receive routine, be called when packet arrives on
network
device */
probe netdev.receive
        =  kernel.function("netif_receive_skb")
{
        dev_name = kernel_string($skb->dev->name)
        length = $skb->len
        protocol = $skb->protocol
        truesize = $skb->truesize
}

/* Queue a buffer for transmission to a network device */
probe netdev.transmit
        =  kernel.function("dev_queue_xmit")
{
        dev_name = kernel_string($skb->dev->name)
        length = $skb->len
        protocol = $skb->protocol
        truesize = $skb->truesize
}
```

Frequently used SystemTap tapsets have been already written.

Tip: To see more tapsets used to trace various sub systems, go to the /usr/share/systemtap/tapset/ directory.

5.1.4 Creating SystemTap scripts

Many SystemTap scripts have already been written to trace various kernel components and kernel subsystems. System analyst has to just run those scripts and analyze the performance bottlenecks. SystemTap scripts can be written to use the tapsets available in the /usr/share/systemtap/tapset directory.

- A sample SystemTap script that uses the networking tapset present in /usr/share/systemtap/tapset/ directory and logs the network data, such as device name, length, true length and protocol during a one second time period, is shown in the Example 5-10.

Example 5-10 Showing a systemtap script that uses networking tapsets for network packet statistics

```
#! /usr/bin/env stap
global cnt, packets

probe netdev.*
{
        printf("dev_name:%s packet_length:%d,
protocol:%d,truesize:%d\n", dev_name, length, protocol, truesize)
        if (length < 100)
                cnt[0]++
        else if (length < 500)
                cnt[1]++
        else if (length < 1000)
                cnt[2]++
        else
                cnt[3]++

        packets++
}

probe timer.ms(1000) { exit () }

probe end {
printf("Total number of packets received and transmitted : %d  \n",
packets)
printf("Number of Packets of size < 100 : %d\n", cnt[0])
printf("Number of Packets of size 100-499 : %d\n", cnt[1])
printf("Number of Packets of size 500-999: %d\n", cnt[2])
printf("Number of Packets of size >= 1000 : %d\n", cnt[3])

}
```

The output of the **stap** command to collect the network statistics for a period of one second is shown in the Example 5-11. The timer probe "probe timers.ms(1000)" ensures that the program runs for 1 second and then calculates the statistics. This script collects the packets transmitted as well recieved through the network card. You can see the total number of packets received and transmitted are 348. The number of packets whose size is less than 100 bytes is 143, the number of packets whose size is less than 500 and greater than 100 bytes is 102, the number of packets of size greater than 500 and less than 1000 are 61 and number of packets greater than 1000 is 42.

Example 5-11 Showing the output of stap to collect network packet statistics

```
#./networking.stp
dev_name:eth0 packet_lenght:66, protocol:2048,truesize:528
dev_name:eth0 packet_lenght:1514, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:650, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:52, protocol:2048,truesize:784
dev_name:eth0 packet_lenght:658, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:52, protocol:2048,truesize:784
dev_name:eth0 packet_lenght:226, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:52, protocol:2048,truesize:784
dev_name:eth0 packet_lenght:226, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:52, protocol:2048,truesize:784
dev_name:eth0 packet_lenght:226, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:52, protocol:2048,truesize:784
dev_name:eth0 packet_lenght:226, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:214, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:52, protocol:2048,truesize:784
Total number of packets received and transmitted : 348
Number of Packets of size < 100 : 143
Number of Packets of size 100-499 : 102
Number of Packets of size 500-999 : 61
Number of Packets of size >= 1000 : 42
```

- The SystemTap script shown in Example 5-11 on page 98 can be modified to show statistics for transmitted and received network packets, as shown in the Example 5-12.

Example 5-12 Showing the script to trace total transmitted and received network packets

```
#! /usr/bin/env stap

global packets, cnt, tpackets, tcnt,

probe netdev.receive {
        printf("dev_name:%s packet_length:%d,
protocol:%d,truesize:%d\n", dev_name, length, protocol, truesize)
        if (length < 100)
                cnt[0]++
        else if (length < 500)
                cnt[1]++
        else if (length < 1000)
                cnt[2]++
        else
                cnt[3]++

        packets++
}

probe netdev.transmit {
        printf("dev_name:%s packet_length:%d,
protocol:%d,truesize:%d\n", dev_name, length, protocol, truesize)
      if (length < 100)
                tcnt[0]++
        else if (length < 500)
                tcnt[1]++
        else if (length < 1000)
                tcnt[2]++
        else
                tcnt[3]++
        tpackets++
}
probe timer.ms(1000) { exit () }

probe end {
printf("Total packets Received: %d  \n", packets)
printf("Number of Packets received of size < 100 : %d\n", cnt[0])
printf("Number of Packets received of size 100-499 : %d\n", cnt[1])
printf("Number of Packets received of size 500-999 : %d\n",
```

```
cnt[2])
printf("Number of Packets received of size >= 1000 : %d\n", cnt[3])

printf("Total packets transmitted : %d  \n", tpackets)
printf("Number of Packets transmitted of size < 100 : %d\n", tcnt[0])
printf("Number of Packets transmitted of size 100-499: %d\n",
tcnt[1])
printf("Number of Packets transmitted of size 500-999 : %d\n",
tcnt[2])
printf("Number of Packets transmitted of size >= 1000 : %d\n", tcnt[3])
}
```

The total number of packets transmitted and received is shown in the Example 5-13. You can see that the total number of packets transmitted are 2415 and the total number of packets received is 2652. You can also see the number of packets with different size transmitted and received.

Example 5-13 Showing the output of stap to trace network packets transmitted and received

```
#./networking.stp
dev_name:eth0 packet_lenght:1500, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:1320, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:66, protocol:2048,truesize:528
dev_name:eth0 packet_lenght:1500, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:1500, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:66, protocol:2048,truesize:528
dev_name:eth0 packet_lenght:1320, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:1500, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:66, protocol:2048,truesize:528
dev_name:eth0 packet_lenght:1500, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:1320, protocol:2048,truesize:2320
dev_name:eth0 packet_lenght:66, protocol:2048,truesize:528
dev_name:eth0 packet_lenght:105, protocol:2048,truesize:2064
dev_name:eth0 packet_lenght:66, protocol:2048,truesize:528
dev_name:eth0 packet_lenght:106, protocol:2048,truesize:784
Total packets Received: 2652
Number of Packets received of size < 100 : 404
Number of Packets received of size 100-499 : 304
Number of Packets received of size 500-999 : 21
Number of Packets received of size >= 1000 : 1923
Total packets transmitted : 2415
Number of Packets transmitted of size < 100 : 859
Number of Packets transmitted of size 100-499 : 746
Number of Packets transmitted of size 500-999 : 21
```

```
Number of Packets transmitted of size >= 1000 : 789
```

A SystemTap script to list the usage of different I/O schedulers for a period of five seconds is shown in the Example 5-14.

Example 5-14 Shows the systemtap script to list the number of requests processed by the I/O scheduler

```
#! /usr/bin/env stap

global cnt
probe ioscheduler. *{
        printf ("elevator_name %s, disk_major %d, disk_minor %d\n",
elevator_name, disk_major, disk_minor)
        if (elevator_name == "cfq")
                cnt[0]++
        else if (elevator_name == "anticipatory")
                cnt[1]++
        else if (elevator_name == "noop")
                cnt[2]++
        else if (elevator_name == "deadline")
                cnt[3]++
        else
                cnt[4]++
}

probe timer.ms(5000) {
        printf(" No of disk I/O scheduled by cfq         : %d\n",
cnt[0])
        printf(" No of disk I/O scheduled by anticipatory: %d\n",
cnt[1])
        printf(" No of disk I/O scheduled by noop        : %d\n",
cnt[2])
        printf(" No of disk I/O scheduled by deadline    : %d\n",
cnt[3])
        exit ()
}
```

The output of the **stap** command to list the number of I/O requests processed by the different I/O scheduler is shown in the Example 5-15. You can also see that all the 449 block I/O requests were processed by cfq scheduler.

Example 5-15 Shows the output of the stap script to list block I/O requests

```
#./iosched.stp
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 8, disk_minor 16
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 8, disk_minor 0
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 8, disk_minor 16
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 8, disk_minor 16
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 16, disk_minor 16
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 8, disk_minor 16
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 8, disk_minor 16
elevator_name cfq, disk_major -1, disk_minor -1
elevator_name cfq, disk_major 16, disk_minor 16
elevator_name cfq, disk_major -1, disk_minor -1
 No of disk I/O scheduled by cfq        : 449
 No of disk I/O scheduled by anticipatory: 0
 No of disk I/O scheduled by noop       : 0
 No of disk I/O scheduled by deadline   : 0
```

The SystemTap script to trace all block I/O requests being made on different disks is shown in the Example 5-16.

Example 5-16 Shows the script to list the number of requests specific to block device

```
#! /usr/bin/env stap

global cnt
probe ioblock.* {
        printf (" %d %s sector %d  %s\n", rw, rw_string, sector
,bio_devname)
        if (bio_devname == "dm-0")
                cnt[0]++
        else if (bio_devname == "sda")
                cnt[1]++
        else if (bio_devname == "sdb")
                cnt[2]++
        else
                cnt[3]++
}

probe timer.ms(5000) {
        printf("Total block I/O to dm-0:         %d\n",cnt[0])
        printf("Total block I/Oto sda:           %d\n",cnt[1])
        printf("Total block I/O to sdb:          %d\n",cnt[2])
        printf("Total block I/O to other device: %d\n",cnt[3])
        printf("Total block I/O:                %d \n",cnt[0] + cnt[1]
+ cnt[2] + cnt[3])       exit ()
}
```

The output of the **stap** command to trace block I/O requests issued and completed is shown in the Example 5-17. You can also see that the total block I/O requests issued to dm-0 is 1167, sda is 913, sdb is 129, and the total block I/O requests issued are 2209.

Example 5-17 Shows the ouput to list the number of requests specific to the block device

```
#./ioblock.stp
1 WRITE sector 27768  dm-0
 1 WRITE sector 28223  sdb
 1 WRITE sector 27776  dm-0
 1 WRITE sector 28231  sdb
 1 WRITE sector 27784  dm-0
 1 WRITE sector 28239  sdb
 1 WRITE sector 27792  dm-0
 1 WRITE sector 28247  sdb
 1 WRITE sector 27800  dm-0
 1 WRITE sector 28255  sdb
 1 WRITE sector 27808  dm-0
 1 WRITE sector 28263  sdb
 1 WRITE sector 27816  dm-0
 1 WRITE sector 28271  sdb
 1 WRITE sector 27824  dm-0
 1 WRITE sector 27832  dm-0
 1 WRITE sector 28279  sdb
 1 WRITE sector 27832  dm-0
Total block I/O to dm-0:         1167
Total block I/Oto sda:           913
Total block I/O to sdb:          129
Total block I/O to other device: 0
Total block I/O:                 2209
```

System analysts can write their own tapsets and run them through the system for performance analysis. Refer to the SystemTap Web site for details to write SystemTap scripts at:

http://sources.redhat.com/systemtap/documentation.html

5.2 CPU utilization monitoring tools

An enterprise application's performance solely depends on the resources available for that application. CPU utilization is the most important resource. Hence, monitoring the CPU utilization across partitions plays an important role in tuning the system. Several monitoring tools available to monitor CPUs across partitions are listed in the following sections.

5.2.1 uptime

This command is used during client crisis situations, where the application's performance is not up to the expectations. The **uptime** command gives the system work load averages from the past one, five, and 15 minutes. Also, it gives the current time, how long the system has been running, and how many users are waiting to be processed.

If the CPU intensive processes are blocked, the load average will increase. On the other hand, if each process gets immediate access to CPU time and there are no CPU cycles lost, the load will decrease.

The optimal value of the load is one, which means that each process has immediate access to the CPU and there are no CPU cycles lost. The typical load can vary from system to system; for uniprocessor workstations, one or two might be acceptable, while for multiprocessor servers, 8 to 10 might be acceptable.

The **uptime** command can be used to determine if the application slow performance is due to CPU intensive or due to other subsystems. For example, consider a network application that is running poorly, and if the system load shown by **uptime** is not high, the problem is more likely to be related to the network rather than the server. The the **uptime** data collected on LPAR configured with one virtual processor and running Jave Data Base Connectivity on DB2 is shown in Example 5-18. You can see the average load on the processor has increased from 4.75 to 7.53. This example shows that the CPU is a bottleneck.

Example 5-18 Shows output of uptime on system running DB2

```
1:00pm  up  19:59,  7 users,  load average: 7.53, 5.72, 4.75
```

Tip: The **/usr/bin/w** command also provides information about who is currently logged on to the machine and what the user is doing. The **/usr/bin/w** command can be used instead of **uptime**.

5.2.2 top

The **top** command provides data about the processor activity in real time. It displays a listing of the most CPU intensive tasks on the system and the most CPU bound applications can be identified using **top**. The output of **top** on a LPAR configured with one virtual processor and running Java database connectivity on DB2 is shown in the Figure 5-1. You can see that the CPU utilization for user processes is 64%, higher than the system utilization, which is 13%. Also, CPU utilization is high for the processes db2fmp, java, db2sync,hald, gnome-terminal, and init.

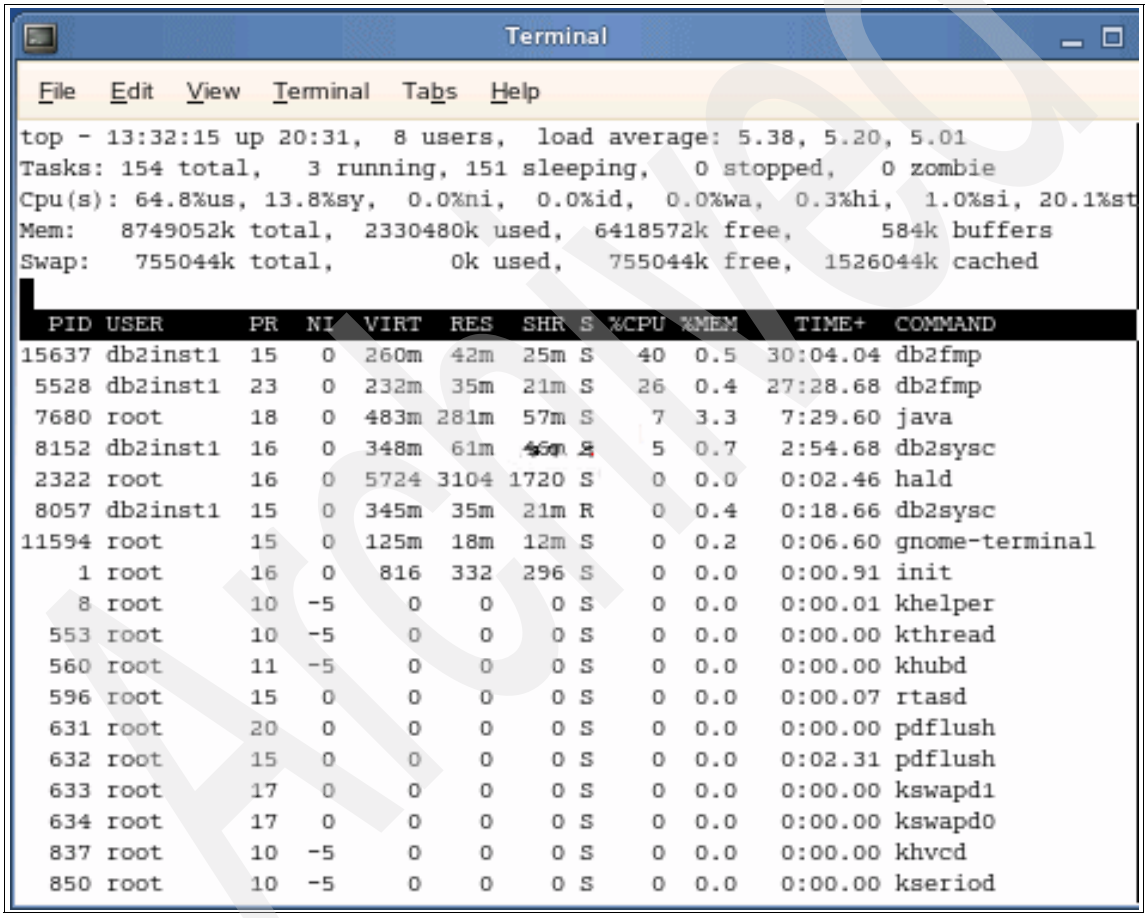


Figure 5-1 Shows the output of the top command while running JDBC™ on DB2 partition.

The columns in the output are as follows:

- PID: Process identification.
- USER: Name of the user who owns (and perhaps started) the process.
- PRI: Priority of the process.
- NI: Niceness level (that is, whether the process tries to be nice by adjusting the priority by the number given).
- SIZE: Amount of memory (code+data+stack), in KB, being used by the process.
- RSS: Amount of physical RAM used, in KB.
- SHARE: Amount of memory shared with other processes, in KB.
- STAT: State of the process: S=sleeping, R=running, T=stopped or traced, D=interruptible sleep, and Z=zombie.
- %CPU: Share of the CPU usage (since the last window update).
- %MEM: Share of physical memory.
- TIME: Total CPU time used by the process (since it was started).
- COMMAND: Command line used to start the task (including parameters).

Tip: You can monitor the CPU utilization for the individual processes using **`#top -p <pid>,<pid>,<pid>`**. The **`/bin/ps`** command gives you a snapshot view of the current processes.

5.2.3 nmon

The **`nmon`** tool is used for monitoring the Linux performance and analyzing performance data, including:

- CPU utilization
- Memory use
- Kernel statistics and run queue information
- Disks I/O rates, transfers, and read/write ratios
- Free space on file systems
- Disk adapters
- Network I/O rates, transfers, and read/write ratios
- Paging space and paging rates
- Top processors

- IBM HTTP Web cache
- User-defined disk groups
- Machine details and resources
- Network File System (NFS)
- Dynamic LPAR (DLPAR) changes: only System p p5 and OpenPower for Linux

Attention: Use **nmon** at your own risk.This **nmon** tool is *NOT OFFICIALLY SUPPORTED*. No warranty is given or implied and you cannot obtain help with it from IBM.

Tip: The **nmon** command is not part of default installation. You need to download and install **nmon** from:

http://www-941.haw.ibm.com/collaboration/wiki/display/WikiPtype/nmon

- The CPU utilization, disk I/O activites, and memory statistics on a LPAR configured with two CPU is shown in the Figure 5-2, where the average CPU utilization is around 55%, free memory is 55% and disk I/O is 100%.

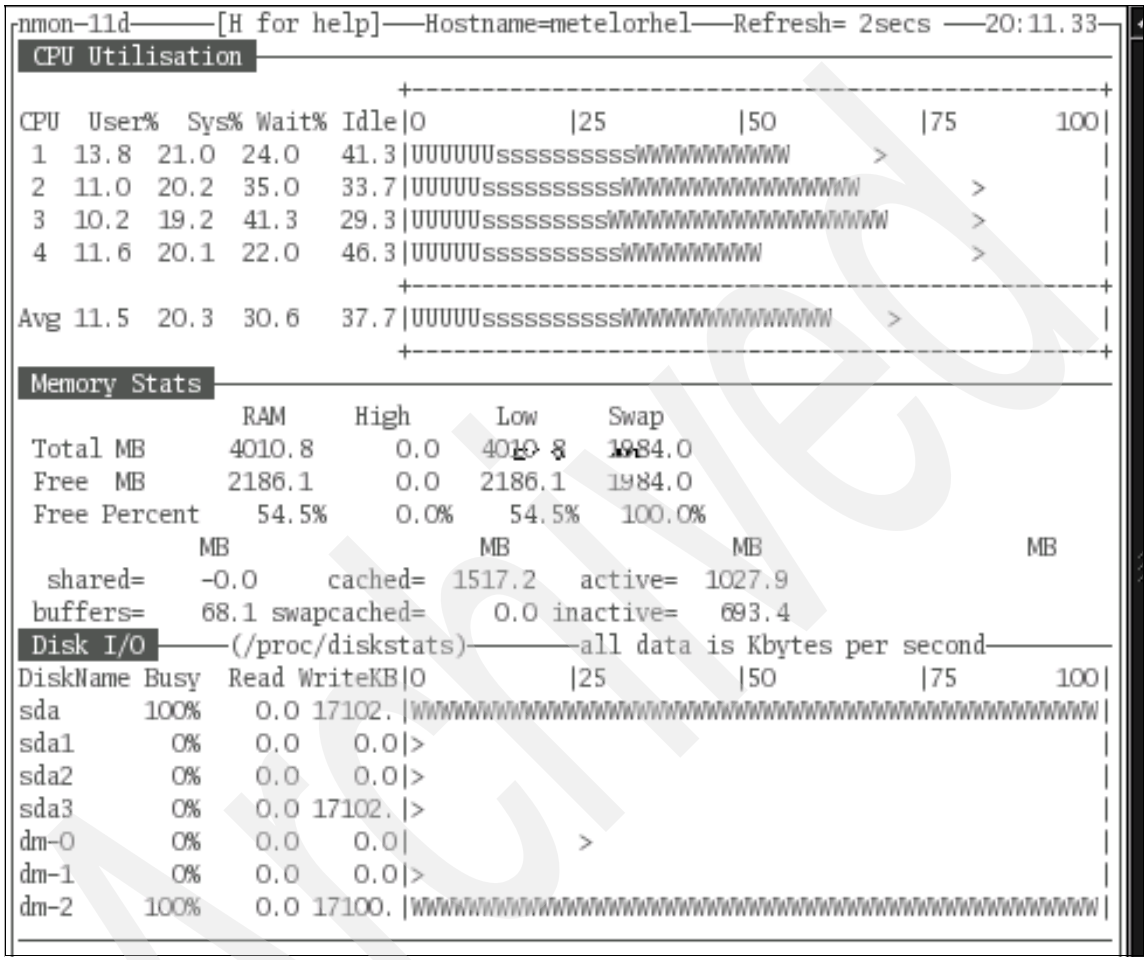


Figure 5-2 Example of nmon output

5.2.4 mpstat

This command displays the activities for each available processor and also global average activities among all processors. The **mpstat** command can be used both in multiprocessor and uniprocessor machines, but on uniprocessor machines it only gives global average activities.

Tip: This command is not part of the default installation**.** You need to install sysstat-5.0.5.11.rhel4.rpm on Red Hat Enterprise Linux AS 4 or sysstat-6.0.2-16.4.ppc.rpm on SUSE Linux Enterprise Server 10 to use the **mpstat** command.

To display three entries of statistics for all processors of a multiprocessor server at 60 seconds intervals, use the following command:

```
#mpstat 60 3 -P ALL
```

The output of the **mpstat** command on an LPAR configured with two cores on SUSE Linux Enterprise Server 10 running Apache with PHP scripts is shown in Example 5-19, where the CPU utilized by the user application is close to 97% and the CPU utilized by the operating system is around 2.17%.

Example 5-19 Showing the output of mpstat on the LPAR partition configured with two cores

```
#mpstat 60 3 -P ALL
Linux 2.6.16.21-0.8-ppc64 (k-sles)      09/25/06

10:56:03     CPU   %user   %nice    %sys %iowait    %irq   %soft %steal %idle intr/s
10:57:03     all   97.50    0.00    2.05    0.00    0.05    0.12 0.00 0.28 166.64
10:57:03       0   97.46    0.00    2.04    0.00    0.03    0.13 0.00 0.33 80.03
10:57:03       1   97.83    0.00    2.07    0.00    0.03    0.13 0.00 0.27 86.61

10:57:03     CPU   %user   %nice    %sys %iowait    %irq   %soft %steal %idle intr/s
10:58:03     all   97.33    0.00    2.12    0.02    0.05    0.12 0.00 0.37 171.71
10:58:03   0 0   97.30    0.00    2.10    0.03    0.07    0.10 0.00 0.40 84.67
10:58:03       1   97.43    0.00    2.10    0.00    0.03    0.10 0.00 0.33 87.07

10:58:03     CPU   %user   %nice    %sys %iowait    %irq   %soft %steal %idle intr/s
10:59:03     all   97.18    0.00    2.18    0.02    0.03    0.12 0.00 0.47 177.23
10:59:03       0   97.31    0.00    2.16    0.00    0.03    0.13 0.00 0.37 89.88
10:59:03       1   96.18    0.00    2.19    0.03    0.07    0.13 0.00 0.53 87.32

Average:     CPU   %user   %nice    %sys %iowait    %irq   %soft %steal %idle intr/s
Average:     all   97.34    0.00    2.12    0.01    0.04    0.12 0.00 0.37 171.87
Average:       0   97.36    0.00    2.10    0.01    0.04    0.12 0.00 0.37 84.87
Average:       1   97.15    0.00    2.12    0.01    0.04    0.12 0.00 0.38 87.00
```

5.3 Memory monitoring tools

Linux virtual memory is a technique that allows applications to run without being completely loaded into the memory.The term virtual memory refers to the abstraction of separating logical memory, as seen by the process, from physical memory, as seen by the CPU. It uses a hard disk as an extension of RAM (known as *swap space*) so that the effective size of usable memory grows correspondingly. The kernel will write the contents of a currently unused block of memory to the hard disk so that the memory can be used for another purpose. When the original contents are needed again, they are read back into memory. This is all made completely transparent to the user; programs running under Linux only see the larger amount of memory available and do not notice that parts of them reside on the disk from time to time.

Tools available to monitor memory utilized by both kernel and applications are discussed in the following sections.

5.3.1 vmstat

The **vmstat** command reports information about the processes, memory, paging. block I/O, traps, and CPU activity. The output of **vmstat** on LPAR configured with two CPUs executing server disk I/O activities is shown in the Example 5-20.

Example 5-20 Example output from vmstat

```
# vmstat
procs -----------memory---------- ---swap-- -----io---- --system--
----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us
sy
id wa
 1  0      0 2484064  90204 1281324    0    0    37   197  519   574  2
7 90  1
```

The columns in the output are as follows:

- Process
  - r: The number of processes waiting for runtime.
  - b: The number of processes in uninterruptible sleep.
- Memory
  - swpd: The amount of virtual memory used.
  - free: The amount of idle memory.
  - buff: The amount of memory used as buffers.
  - cache: The amount of memory used as cache.
  - inact: The amount of inactive memory. (-a option)

  - active: The amount of active memory. (-a option)
- Swap
  - si: Amount of memory swapped from the disk (KBps).
  - so: Amount of memory swapped to the disk (KBps).
- IO
  - bi: Blocks sent to a block device (blocks/s).
  - bo: Blocks received from a block device (blocks/s).
- System
  - in: The number of interrupts per second, including the clock.
  - cs: The number of context switches per second.
- CPU (these are percentages of total CPU time)
  - us: Time spent running non-kernel code (user time, including nice time).
  - sy: Time spent running kernel code (system time).
  - id: Time spent idle. Prior to Linux 2.5.41, this included IO-wait time.
  - wa: Time spent waiting for IO. Prior to Linux 2.5.41, this appeared as zero.

5.3.2 top

The **top** command can be also used to get per-process virtual memory usage information.

For more details about the **top** command, refer to 5.2.2, “top” on page 106.

5.3.3 pmap

The **pmap** command provides information about a process’s memory mappings, such as its stack, data segment, mapped files, and so on.

The result of the command on an LPAR configured with two CPUs, running **smdb** on Red Hat Enterprise Linux AS 4 update 4, is shown in the Example 5-21

```
#pmap -x <pid>
```

Example 5-21 Showing output of pmap for smbd process

```
#pmap -x 3319
3319:   smbd -D
Address   Kbytes     RSS    Anon  Locked Mode   Mapping
078b2000      44       -       -       - r-x--  libnss_files-2.3.4.so
078bd000      60       -       -       - -----  libnss_files-2.3.4.so
078cc000       4       -       -       - r----  libnss_files-2.3.4.so
078cd000       4       -       -       - rwx--  libnss_files-2.3.4.so
078de000       8       -       -       - r-x--  IBM850.so
078e0000      60       -       -       - -----  IBM850.so
078ef000       4       -       -       - r----  IBM850.so
078f0000       4       -       -       - rwx--  IBM850.so
```

```
07901000      12        -        -        - r-x--  UTF-16.so
07904000      60        -        -        - -----  UTF-16.so
07913000       4        -        -        - r----  UTF-16.so
07914000       4        -        -        - rwx--  UTF-16.so
07925000      64        -        -        - r-x--  libaudit.so.0.0.0
07935000      60        -        -        - -----  libaudit.so.0.0.0
07944000       4        -        -        - r----  libaudit.so.0.0.0
07945000       4        -        -        - rwx--  libaudit.so.0.0.0
07956000      72        -        -        - r-x--  libz.so.1.2.1.2
07968000      60        -        -        - -----  libz.so.1.2.1.2
07977000       4        -        -        - rwx--  libz.so.1.2.1.2
07988000      92        -        -        - r-x--  libsasl2.so.2.0.19
0799f000      64        -        -        - -----  libsasl2.so.2.0.19
079af000       4        -        -        - rwx--  libsasl2.so.2.0.19
079c0000    1240        -        -        - r-x--  libc-2.3.4.so
07af6000      60        -        -        - -----  libc-2.3.4.so
07b05000       4        -        -        - r----  libc-2.3.4.so
07b06000      20        -        -        - rwx--  libc-2.3.4.so
07b0b000       8        -        -        - rwx--    [ anon ]
07b1d000      36        -        -        - r-x--  libpopt.so.0.0.0
07b26000      60        -        -        - -----  libpopt.so.0.0.0
07b35000       4        -        -        - rwx--  libpopt.so.0.0.0
07b46000       8        -        -        - r-x--  libdl-2.3.4.so
07b48000      64        -        -        - -----  libdl-2.3.4.so
07b58000       4        -        -        - r----  libdl-2.3.4.so
07b59000       4        -        -        - rwx--  libdl-2.3.4.so
07b6a000      32        -        -        - r-x--  libacl.so.1.1.0
07b72000      60        -        -        - -----  libacl.so.1.1.0
07b81000       4        -        -        - rwx--  libacl.so.1.1.0
07b92000      16        -        -        - r-x--  libattr.so.1.1.0
07b96000      60        -        -        - -----  libattr.so.1.1.0
07ba5000       4        -        -        - rwx--  libattr.so.1.1.0
07bb6000      40        -        -        - r-x--  libpam.so.0.77
07bc0000      60        -        -        - -----  libpam.so.0.77
07bcf000       4        -        -        - rwx--  libpam.so.0.77
07be0000      20        -        -        - r-x--  libcrypt-2.3.4.so
07be5000      60        -        -        - -----  libcrypt-2.3.4.so
07bf4000       4        -        -        - r----  libcrypt-2.3.4.so
07bf5000       4        -        -        - rwx--  libcrypt-2.3.4.so
07bf6000     156        -        -        - rwx--    [ anon ]
07c2d000      80        -        -        - r-x--  libnsl-2.3.4.so
07c41000      60        -        -        - -----  libnsl-2.3.4.so
07c50000       4        -        -        - r----  libnsl-2.3.4.so
07c51000       4        -        -        - rwx--  libnsl-2.3.4.so
07c52000       8        -        -        - rwx--    [ anon ]
```

```
07c64000    1024       -       -       - r-x--  libcrypto.so.0.9.7a
07d64000      76       -       -       - rwx--  libcrypto.so.0.9.7a
07d77000      12       -       -       - rwx--    [ anon ]
07d8a000     208       -       -       - r-x--  libssl.so.0.9.7a
07dbe000      64       -       -       - -----  libssl.so.0.9.7a
07dce000      16       -       -       - rwx--  libssl.so.0.9.7a
07de2000     112       -       -       - r-x--  libcups.so.2
07dfe000      60       -       -       - -----  libcups.so.2
07e0d000      12       -       -       - rwx--  libcups.so.2
07e20000      68       -       -       - r-x--  libresolv-2.3.4.so
07e31000      60       -       -       - -----  libresolv-2.3.4.so
07e40000       4       -       -       - r----  libresolv-2.3.4.so
07e41000       4       -       -       - rwx--  libresolv-2.3.4.so
07e42000       8       -       -       - rwx--    [ anon ]
07e54000      12       -       -       - r-x--  libcom_err.so.2.1
07e57000      60       -       -       - -----  libcom_err.so.2.1
07e66000       4       -       -       - rwx--  libcom_err.so.2.1
07e77000     136       -       -       - r-x--  libk5crypto.so.3.0
07e99000      60       -       -       - -----  libk5crypto.so.3.0
07ea8000       8       -       -       - rwx--  libk5crypto.so.3.0
07eba000     412       -       -       - r-x--  libkrb5.so.3.2
07f21000      60       -       -       - -----  libkrb5.so.3.2
07f30000      16       -       -       - rwx--  libkrb5.so.3.2
07f44000      80       -       -       - r-x--  libgssapi_krb5.so.2.2
07f58000      64       -       -       - -----  libgssapi_krb5.so.2.2
07f68000       4       -       -       - rwx--  libgssapi_krb5.so.2.2
07f79000      56       -       -       - r-x--  liblber-2.2.so.7.0.6
07f87000      64       -       -       - -----  liblber-2.2.so.7.0.6
07f97000       4       -       -       - rwx--  liblber-2.2.so.7.0.6
07fa8000     212       -       -       - r-x--  libldap-2.2.so.7.0.6
07fdd000      64       -       -       - -----  libldap-2.2.so.7.0.6
07fed000      12       -       -       - rwx--  libldap-2.2.so.7.0.6
08000000    2916       -       -       - r-x--  smbd
082e9000     100       -       -       - rwx--  smbd
08302000     824       -       -       - rwx--    [ anon ]
f7892000    4160       -       -       - rw-s-  locking.tdb
f7ca2000      24       -       -       - rw-s-  sessionid.tdb
f7ca8000     784       -       -       - rw---    [ anon ]
f7d6e000      64       -       -       - r--s-  valid.dat
f7d7e000      24       -       -       - r--s-  gconv-modules.cache
f7d84000    2048       -       -       - r----  locale-archive
f7f84000     128       -       -       - r--s-  lowcase.dat
f7fa4000      24       -       -       - rw---    [ anon ]
f7faf000       8       -       -       - rw-s-  account_policy.tdb
f7fb1000       8       -       -       - rw-s-  secrets.tdb
```

```
f7fb3000       8       -       -       - rw-s-  group_mapping.tdb
f7fb5000       8       -       -       - rw-s-  ntdrivers.tdb
f7fb7000     128       -       -       - r--s-  upcase.dat
f7fd7000      92       -       -       - r-x--  ld-2.3.4.so
f7fef000       8       -       -       - rw-s-  ntprinters.tdb
f7ff1000       8       -       -       - rw-s-  connections.tdb
f7ff3000       8       -       -       - rw-s-  registry.tdb
f7ff5000       4       -       -       - rw-s-  gencache.tdb
f7ff6000       8       -       -       - rw-s-  share_info.tdb
f7ff9000       4       -       -       - rw-s-  brlock.tdb
f7ffa000       4       -       -       - rw-s-  ntforms.tdb
f7ffc000       4       -       -       - rw-s-  messages.tdb
f7ffd000       4       -       -       - r----  ld-2.3.4.so
f7ffe000       4       -       -       - rwx--  ld-2.3.4.so
ffff1000      56       -       -       - rw---    [ stack ]
-------- ------- ------- ------- -------
total kB   17404       -       -       -
```

5.3.4 free

The **free** command provides the total amount of free and used physical and swap memory of the system, as well as the buffers used by the kernel. The total, used, free, buffers, and cache memory on the LPAR configured with two CPUs is shown in the Example 5-22, where the total memory is 4.1 MB, and used and free memory is 2 MB.

Example 5-22 Showing the output of the free command to show the memory usage

```
# free
             total       used       free     shared    buffers
cached
Mem:       4107072    2004968    2102104          0      93528
1659840
-/+ buffers/cache:     251600    3855472
Swap:      2031608          0    2031608
```

5.4 Network monitoring tools

Linux network subsystem supports various types of protocols to communicate across the systems. The performance of the network intensive applications can be tuned by monitoring the network subsystem and collecting the relevant information. The tools required to monitor the network system are listed in the following sections.

5.4.1 netstat

The **netstat** command can be used to print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships. The number of IP/UDP packets received, TCP active/passive connections, and TCP segments transmitted or received is shown in the Example 5-23. It also shows that the server has four passive connections, receiving and transmitting huge number of segments:

```
netstat -s: Show network statistics
```

Example 5-23 List network statistics

```
netstat -s: Show network statistics
Ip:
    9516778 total packets received
    0 forwarded
    0 incoming packets discarded
    9516775 incoming packets delivered
    7117173 requests sent out
Icmp:
    0 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
    0 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
Tcp:
    0 active connections openings
    4 passive connection openings
    0 failed connection attempts
    0 connection resets received
    4 connections established
    9516547 segments received
    7117139 segments send out
    0 segments retransmited
    0 bad segments received.
```

```
    0 resets sent
Udp:
    166 packets received
    0 packets to unknown port received.
    0 packet receive errors
    36 packets sent
TcpExt:
    ArpFilter: 0
    60 delayed acks sent
    897439 packets directly queued to recvmsg prequeue.
    529483394 packets directly received from backlog
    855797676 packets directly received from prequeue
    8418261 packets header predicted
    1059051 packets header predicted and directly queued to user
    TCPPureAcks: 25
    TCPHPAcks: 2557347
    TCPRenoRecovery: 0
    TCPSackRecovery: 0
    TCPSACKReneging: 0
    TCPFACKReorder: 0
    TCPSACKReorder: 0
    TCPRenoReorder: 0
    TCPTSReorder: 0
    TCPFullUndo: 0
    TCPPartialUndo: 0
    TCPDSACKUndo: 0
    TCPLossUndo: 0
    TCPLoss: 0
    TCPLostRetransmit: 0
    TCPRenoFailures: 0
    TCPSackFailures: 0
    TCPLossFailures: 0
    TCPFastRetrans: 0
    TCPForwardRetrans: 0
    TCPSlowStartRetrans: 0
    TCPTimeouts: 0
    TCPRenoRecoveryFail: 0
    TCPSackRecoveryFail: 0
    TCPSchedulerFailed: 0
    TCPRcvCollapsed: 0
    TCPDSACKOldSent: 0
    TCPDSACKOfoSent: 0
    TCPDSACKRecv: 0
    TCPDSACKOfoRecv: 0
    TCPAbortOnSyn: 0
```

```
    TCPAbortOnData: 0
    TCPAbortOnClose: 0
    TCPAbortOnMemory: 0
    TCPAbortOnTimeout: 0
    TCPAbortOnLinger: 0
    TCPAbortFailed: 0
    TCPMemoryPressures: 0
```

The kernel interfaces used to transfer and receive the packets is shown in Example 5-24. It shows that eth0 MTU is 1500 and lots of packets are transferred and received through the eth0 interface.

```
#netstat -a -i eth0: Kernel interface table info
```

Example 5-24 Example output shows the kernel interface table info

```
Kernel Interface table
Iface       MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR
TX-DRP TX-OVR Flg
eth0       1500   0 13286447      0      0      0  9927671      0
0      0 BMRU
lo        16436   0        8      0      0      0        8      0
0      0 LRU
sit0       1480   0        0      0      0      0        0      0
0      0 O
```

Tip: More **netstat** commands

```
netstat -nap: List all connected processes
netstat -punta: :Lists all the externally connected processes
```

5.4.2 traceroute

The **traceroute** command can be used to print the route packets taken to network host. Tracking the route one's packets follow (or finding the gateway that is discarding particular packets) can be difficult, since the internet is a large and complex aggregation of network hardware, connected together by gateways. The **traceroute** command utilizes the IP protocol “time to live” field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host.

The gateways 12, 14, 15, 16, and 17 (shown in Example 5-25 on page 119) hops away either do not send ICMP “time exceeded” messages or send them with a ttl too small to reach us. 14 - 17 are running the MIT C Gateway code that does not send “time exceeded” messages.

Example 5-25 Example of traceroute

```
[yak 72]% traceroute allspice.lcs.mit.edu.
             traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops
max
              1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
              2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms
19 ms
              3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms
19 ms
              4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39
ms  39 ms
              5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39
ms
39 ms
              6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
              7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
              8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
              9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
             10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
             11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
             12  * * *
             13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
             14  * * *
             15  * * *
             16  * * *
             17  * * *
             18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms
279 ms
```

5.4.3 iptraf

The **iptraf** command is a network monitoring utility for IP networks. It intercepts packets in the network and gives out various pieces of information about the current IP traffic over it.The **iptraf** tool can be used to monitor the load on an IP network, the most used types of network services, the proceedings of TCP connections, and others.The **iptraf** command utilizes the built-in raw packet capture interface of the Linux kernel, allowing it to be used with a wide range of Ethernet cards, supported FDDI adapters, supported ISDN adapters, Token Ring, asynchronous SLIP/ PPP interfaces, and other network devices. Information returned by **iptraf** includes:

- Total, IP, TCP, UDP, ICMP, and non-IP byte counts
- TCP source and destination addresses and ports

- TCP packet and byte counts
- TCP flag statuses
- UDP source and destination information
- ICMP type information
- OSPF source and destination information
- TCP and UDP service statistics
- Interface packet counts
- Interface IP checksum error counts
- Interface activity indicators
- LAN station statistics

More information about **iptraf** can be found at:

http://iptraf.seul.org/2.7/manual.html

The **iptraf** main dialog to choose the statistical facilities is shown in Figure 5-3.

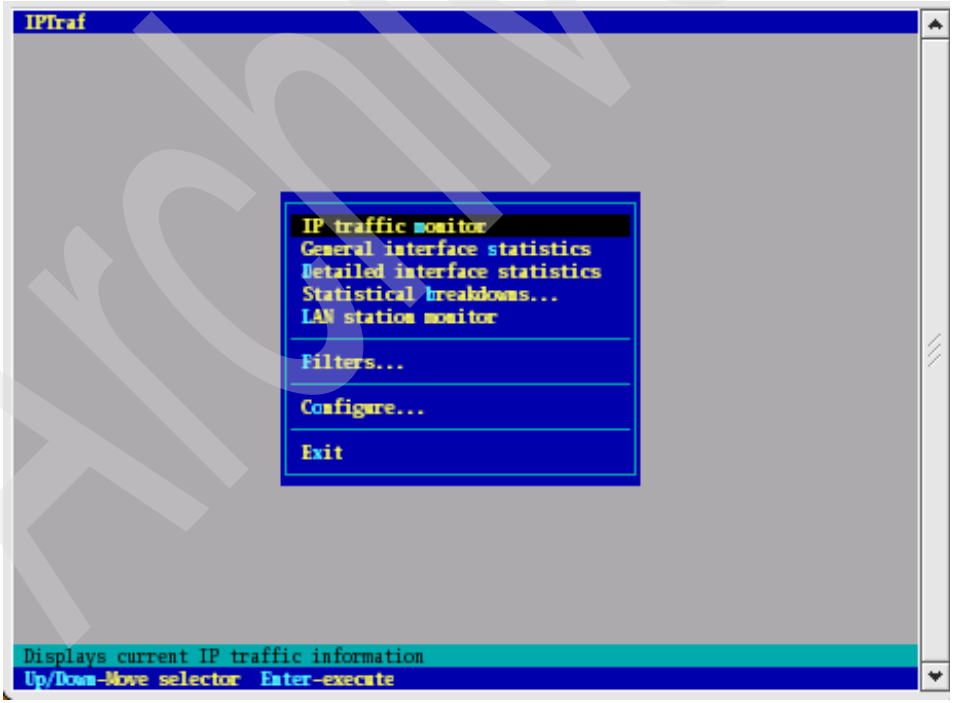


Figure 5-3 Shows the iptraf main dialog

The number of packets received, packet size, source address, flags, interface eth0, and the port for each TCP connection is shown in Figure 5-4.

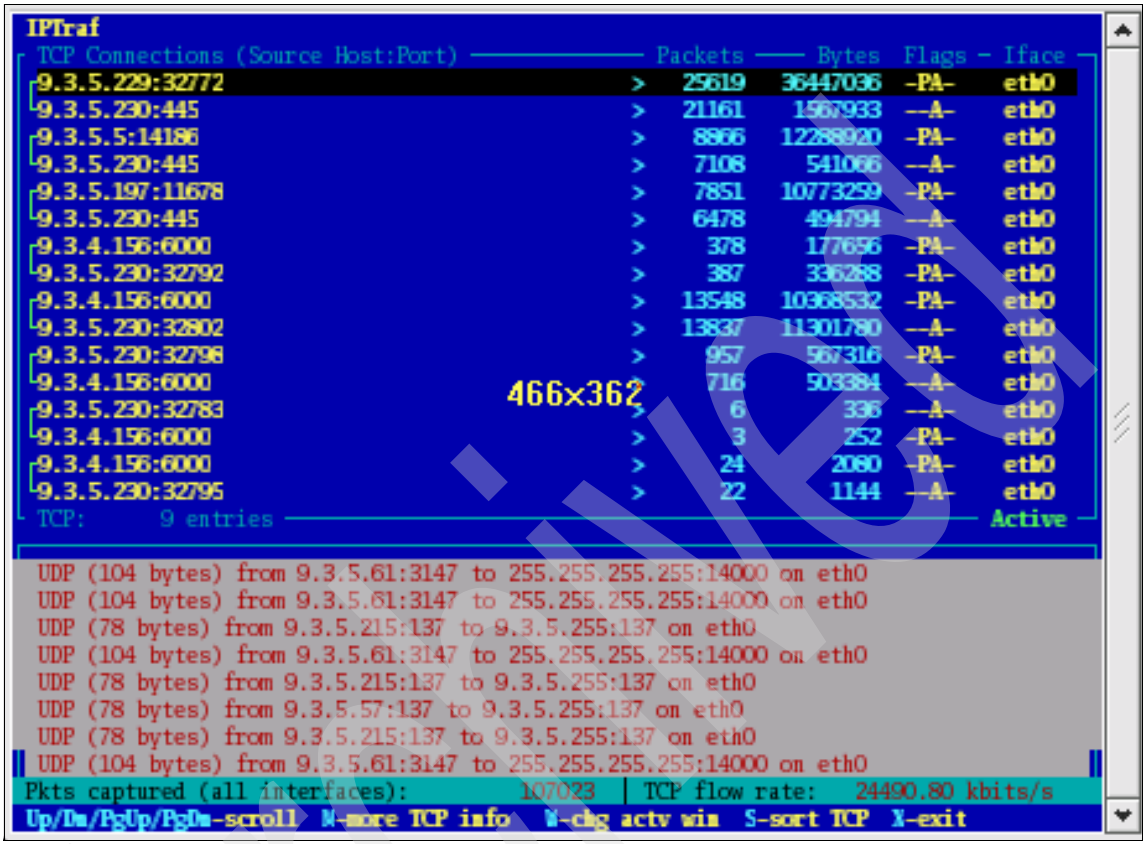


Figure 5-4 Shows IP Traffic Monitor

The packet size statistical breakdown is shown in Figure 5-5. You can see the maximum number of packets transmitted are from the size range of 1 to 150 bytes.

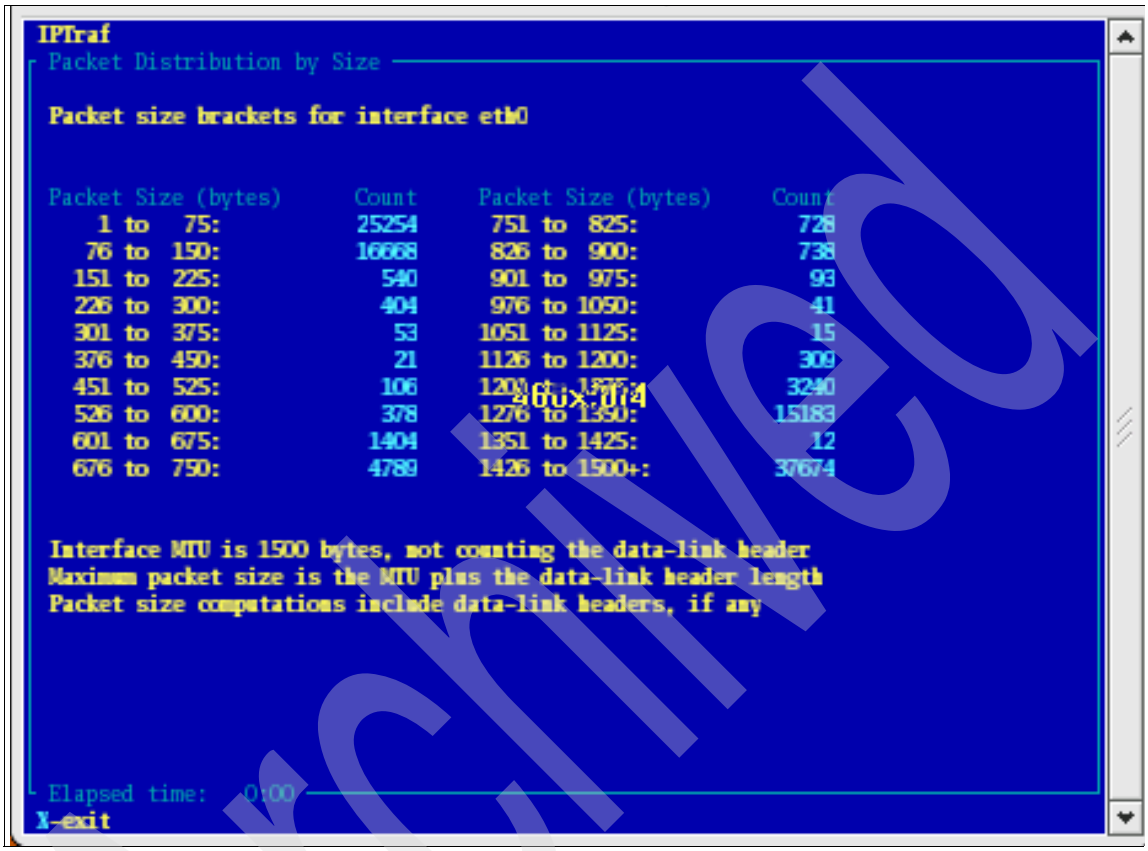


Figure 5-5 Shows the packets size statistical breakdown.

5.4.4 Ethreal

Ethreal is a graphical tool used to capture the data “off the wire” from a live network connection, or even read from a file, and analyses it.

Figure 5-6 shows the Ethereal network analyzer main dialog.

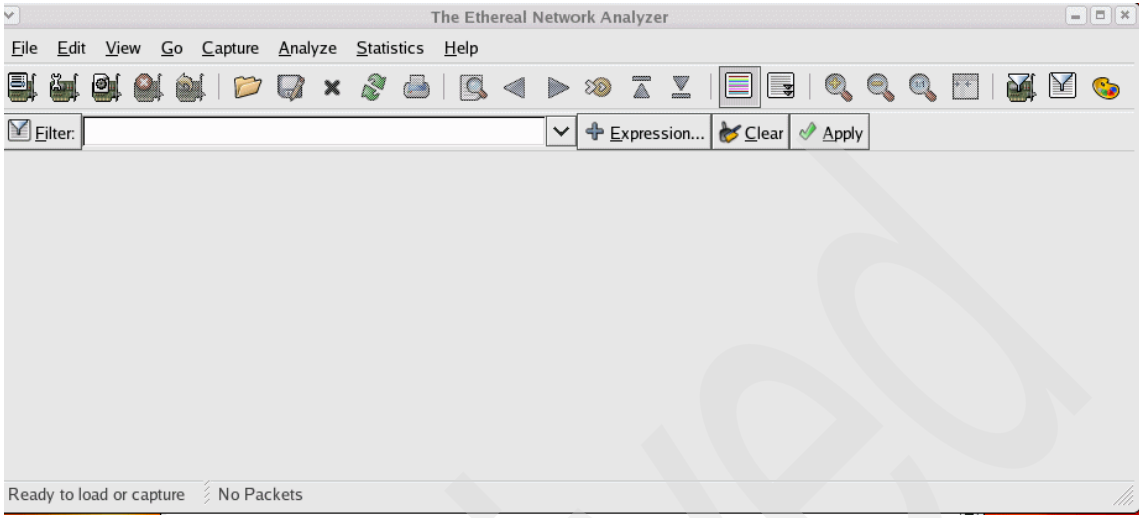


Figure 5-6 Shows the Ethereal Network Analyzer main menu

Now you want to capture some data from the network attached to your workstation. Do this by selecting **Capture** → **start options** and you see the dialog shown in Figure 5-7. In Figure 5-7, we choose to capture the TCP patches transmitted and received on the eth0 interface.

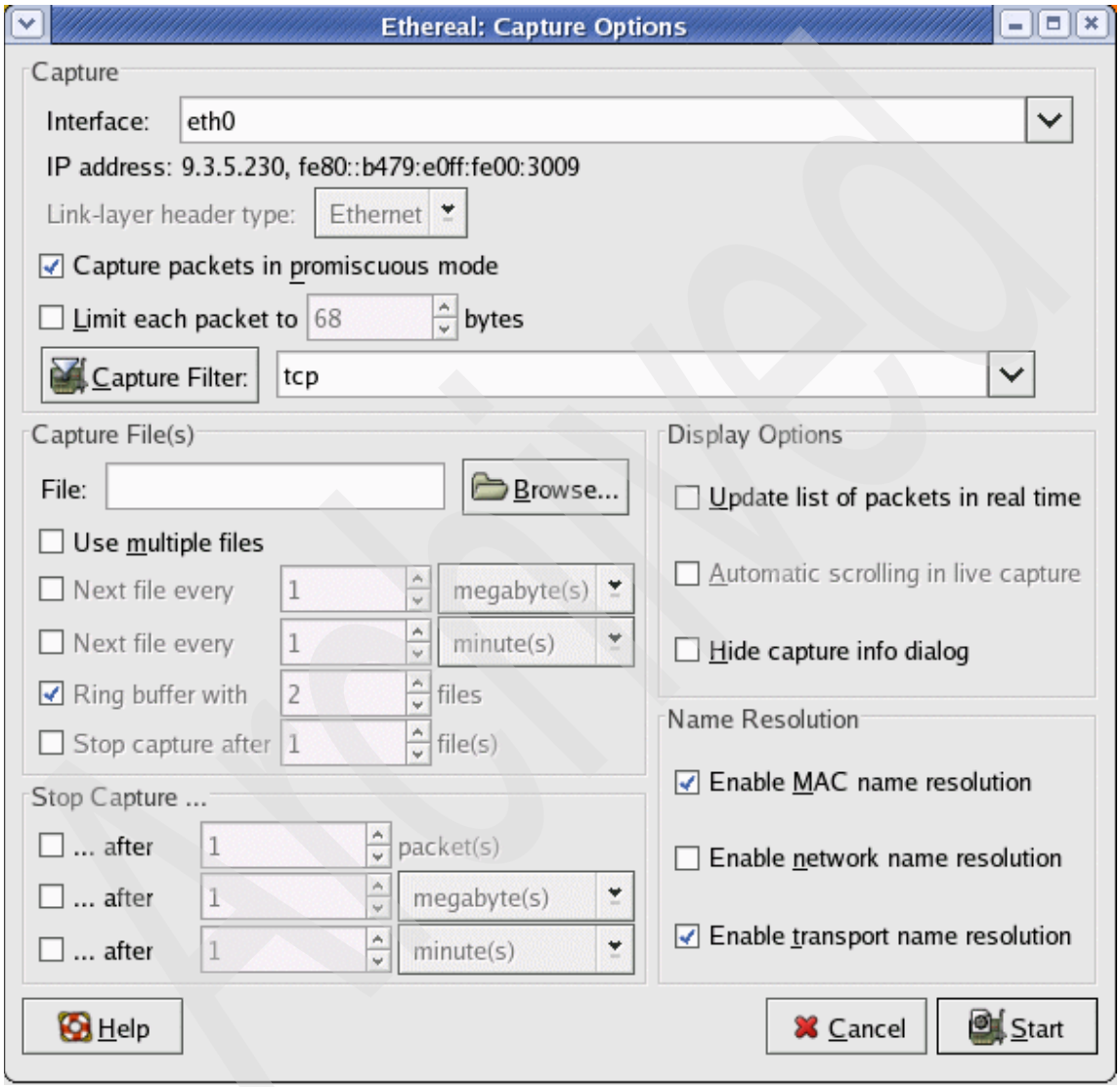


Figure 5-7 Shows the Ethereal Capture options.

The number of packets captured from eth0 interface can be see in Figure 5-8

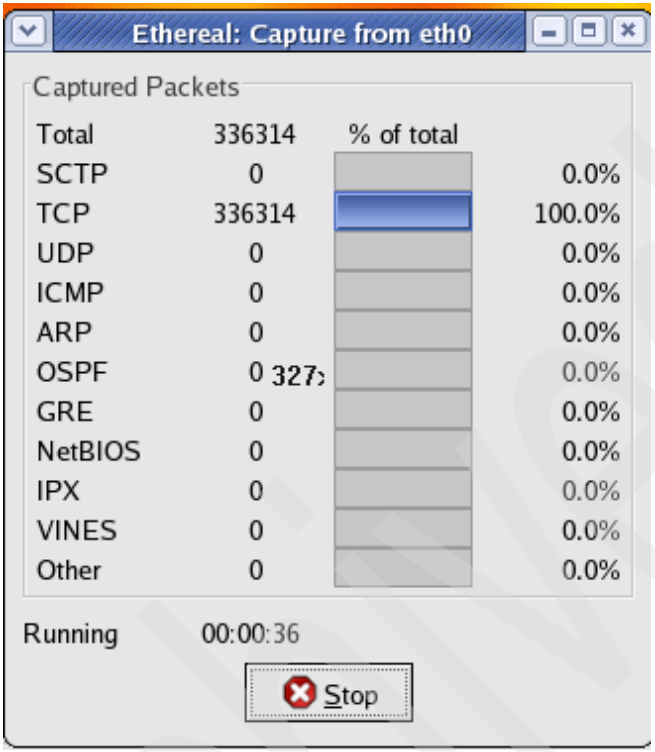


Figure 5-8 Showing the number of TCP packets captured at the eth0 interface

The capture and dissection of all the TCP patches received and transmitted is shown in Figure 5-9. Ethreal automatically highlights the raw bytes corresponding to the selected field.

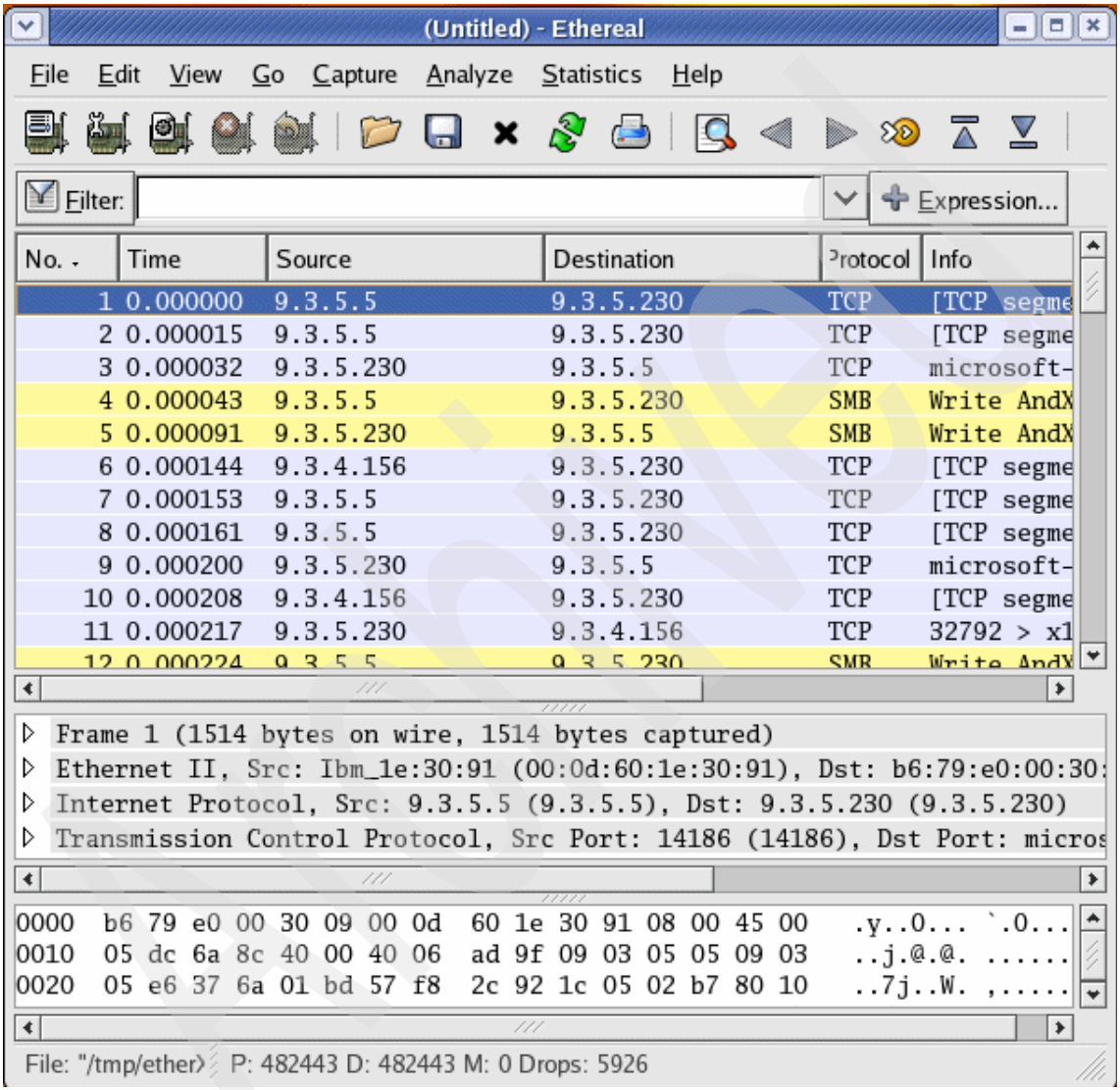


Figure 5-9 Shows the TCP patches captured during monitoring

5.5 Disk I/O monitoring tools

The performance of the disk I/O activity can have a substantial bearing upon the performance and use of the applications. In short, the ability to easily and quickly assess and appreciate, especially in immediate empirical terms, the performance of file I/O activity provides the potential for better understanding of, substantive improvements to, and other benefits directly related to the performance of the associated applications and their use. Faster files can make applications faster, which in turn promotes better productivity, efficiency, and ROI.

5.5.1 iostat

The **iostat** command is used for monitoring system input/output device loading by observing the time the devices are active in relation to their average transfer rates. The **iostat** command generates reports that can be used to change the system configuration to better balance the input/output load between physical disks. The output of **iostat** command on a LPAR partition configured with two CPUs doing server data transfer to and from the server is shown in Example 5-26.

You can see in Example 5-26 that the number of I/Os are intended more for sda, sda3, dm-0, and the dm-2 disk partition.

Example 5-26 Sample output of iostat command

```
# iostat
Linux 2.6.9-42.EL (rhel)  09/25/2006

avg-cpu:  %user   %nice    %sys %iowait   %idle
          14.12    0.14   22.70    1.55   61.49

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda              33.64       232.54      2166.52    2869888   26737884
sda1              0.02         0.03         0.00        388          0
sda2              0.06         0.07         0.00        816          4
sda3            275.07       232.35      2164.97    2867500   26718792
dm-0            115.10       186.06       891.23    2296258   10998984
dm-1              0.01         0.03         0.02        360        288
dm-2            160.14        46.21      1275.27     570338   15738608
```

The CPU utilization report has four sections:

- %user: Shows the percentage of CPU utilization that was taken up while executing at the user level (applications).
- %nice: Shows the percentage of CPU utilization that was taken up while executing at the user level with a nice priority.
- %sys: Shows the percentage of CPU utilization that was taken up while executing at the system level (kernel).
- %idle: Shows the percentage of time the CPU was idle.

The device utilization report is split into the following sections:

- Device: The name of the block device.
- tps: The number of transfers per second (I/O requests per second) to the device. Multiple single I/O requests can be combined in a transfer request, because a transfer request can have different sizes.
- Blk_read/s, Blk_wrtn/s: Blocks read and written per second indicate data read/written from/to the device in seconds. Blocks may also have different sizes. Typical sizes are 1024, 2048, or 4048 bytes, depending on the partition size. For example, the block size of /dev/sda1 can be found with:

  ```
  #dumpe2fs -h /dev/sda1 |grep -F "Block size"
  ```

  which will give an output similar to:

  ```
  dumpe2fs 1.34 (25-Jul-2003)
  Block size:               1024
  ```

- Blk_read, Blk_wrtn: This indicates the total number of blocks read/written since the boot.

Tip: Some more examples of **iostat**:

- **iostat -d 2**: Displays a continuous device report at two second intervals.
- **iostat -d 2 6**: Displays six reports at two second intervals for all devices.
- **iostat -x hda hdb 2 6**: Displays six reports of extended statistics at two second intervals for devices hda and hdb.

5.6 Miscellaneous monitoring tools

A certain set of tools on Linux provides complete system information that can be used to narrow down and identify the performance bottlenecks with the system.

5.6.1 dmesg

The main purpose of the **dmesg** command is to display kernel messages. The **dmesg** command can provide helpful information in case of hardware problems or problems with kernel configuration or kernel module or depleting kernel resources. Linux kernel bootup messages and subsystem initialization messages are also displayed by **dmesg**.

The output of **dmesg** on LPAR configured with eight CPUS is shown in Example 5-27, where the system hardware detection and various configurations, such as memory, disk configuration, and file system, can be seen.

Example 5-27 Partial output from dmesg

```
Found initrd at 0xc000000002300000:0xc0000000024c0c00
firmware_features = 0x1ffd5f
Partition configured for 8 cpus.
Starting Linux PPC64 2.6.9-42.EL
-----------------------------------------------------
naca                         = 0xc000000000004000
naca->pftSize                = 0x1b
naca->debug_switch           = 0x0
naca->interrupt_controller   = 0x2
systemcfg                    = 0xc000000000005000
systemcfg->processorCount    = 0x2
systemcfg->physicalMemorySize = 0x100000000
systemcfg->dCacheL1LineSize  = 0x80
systemcfg->iCacheL1LineSize  = 0x80
htab_data.htab               = 0x0000000000000000
htab_data.num_ptegs          = 0x100000
-----------------------------------------------------
[boot]0100 MM Init
[boot]0100 MM Init Done
Linux version 2.6.9-42.EL (bhcompile@js20-bc1-12.build.redhat.com) (gcc
version 3.4.6 20060404 (Red Hat 3.4.6-2)) #1 SMP Wed Jul 12 23:22:51
EDT 2006
[boot]0012 Setup Arch
Node 0 Memory: 0x0-0xa0000000
Node 1 Memory: 0xa0000000-0x100000000
EEH: No capable adapters found
```

```
PPC64 nvram contains 7168 bytes
Using shared processor idle loop
On node 0 totalpages: 655360
  DMA zone: 655360 pages, LIFO batch:16
  Normal zone: 0 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
On node 1 totalpages: 393216
  DMA zone: 393216 pages, LIFO batch:16
  Normal zone: 0 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
[boot]0015 Setup Done
Built 2 zonelists
Kernel command line: root=/dev/VolGroup00/LogVol00 ro smt-enabled=off
[boot]0020 XICS Init
xics: no ISA interrupt controller
[boot]0021 XICS Done
PID hash table entries: 4096 (order: 12, 131072 bytes)
time_init: decrementer frequency = 188.045000 MHz
time_init: processor frequency   = 1502.496000 MHz
Console: colour dummy device 80x25
Dentry cache hash table entries: 1048576 (order: 11, 8388608 bytes)
Inode-cache hash table entries: 524288 (order: 10, 4194304 bytes)
freeing bootmem node 0
freeing bootmem node 1
Memory: 4104640k/4194304k available (3076k kernel code, 89236k
reserved, 1328k data, 522k bss, 208k init)
Calibrating delay loop... 371.71 BogoMIPS (lpj=185856)

Linux Kernel Card Services
  options:  [pci] [cardbus]
ip_tables: (C) 2000-2002 Netfilter core team
ip_tables: (C) 2000-2002 Netfilter core team
(drivers/net/ibmveth.c:442 ua:30000009): open starting
(drivers/net/ibmveth.c:492 ua:30000009): buffer list @
0xc000000000fb3d000
(drivers/net/ibmveth.c:493 ua:30000009): filter list @
0xc000000004d4c000
(drivers/net/ibmveth.c:494 ua:30000009): receive q   @
0xc0000000073f0000
(drivers/net/ibmveth.c:514 ua:30000009): registering irq 0xbf
(drivers/net/ibmveth.c:525 ua:30000009): initial replenish cycle
(drivers/net/ibmveth.c:530 ua:30000009): open complete
Attached scsi generic sg0 at scsi0, channel 0, id 1, lun 0,  type 0
lp: driver loaded but no devices found
NET: Registered protocol family 10
```

```
Disabled Privacy Extensions on device c000000000455958(lo)
IPv6 over IPv4 tunneling driver
divert: not allocating divert_blk for non-ethernet device sit0
eth0: no IPv6 routers present
kjournald starting.  Commit interval 5 seconds
EXT3 FS on dm-2, internal journal
EXT3-fs: mounted filesystem with writeback data mode.
```

5.6.2 ulimit

This command is built into the bash shell and is used to provide control over the resources available to the shell and to the processes started by it on systems that allow such control.

Use the option -a to list all the parameters that we can set. The output of the **ulimit** command on an LPAR machine configured with two CPUs is shown in Example 5-28.

```
ulimit -a
```

Example 5-28 Output of ulimit

```
# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) 4
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 7168
virtual memory          (kbytes, -v) unlimited
```

The -H and -S options specify the hard and soft limits that can be set for the given resource. If the soft limit is passed, the system administrator will receive a warning. The hard limit is the maximum value that can be reached before the user gets the error message `Out of file handles`.

For example, you can set a hard limit for the number of file handles and open files (-n):

```
#ulimit -Hn 4096
```

For the soft limit of number of file handles and open files, use:

```
#ulimit -Sn 1024
```

To set the hard and soft values, issue the command with a new value:

```
#ulimit -Hn
#ulimit -Sn
```

This command can be used, for example, to limit Oracle users on the fly. To set it on startup, enter the follow lines, for example, in /etc/security/limits.conf:

```
soft nofile 4096
hard nofile 10240
```

In addition, for Red Hat Enterprise Linux AS 4, make sure that the file /etc/pam.d/system-auth has the following entry:

```
session     required      /lib/security/$ISA/pam_limits.so
```

In addition, for SUSE Linux Enterprise Server 10, make sure that the files /etc/pam.d/login and /etc/pam.d/sshd have the following entry:

```
session required       pam_limits.so
```

This entry is required so that the system can enforce these limits.

5.6.3 /proc

/proc is very special in that it is also a virtual file system. It is sometimes referred to as a process information pseudo-file system. It does not contain “real” files but instead contains runtime system information (for example, system memory, devices mounted, hardware configuration, and so on). For this reason, it can be regarded as a control and information center for the kernel. In fact, many system utilities are simply calls to files in this directory. For example, **lsmod** is the same as **cat /proc/modules**, while **lspci** is synonymous with **cat /proc/pci**. By altering files located in this directory, you can even read/change kernel parameters (sysctl) while the system is running. Most of the useful information can be gathered from **/proc**. Commonly used **/proc** entries by the administrators are discussed below:

/proc/ppc64/lparcfg

This file /proc/ppc64/lparcfg contains the partition information. The /proc/ppc64/lparcfg file on LPAR partition configured with two processors is shown in the Example 5-29.

Example 5-29 Sample /proc/ppc64/lparcfg

```
# cat /proc/ppc64/lparcfg
lparcfg 1.6
serial_number=IBM,0210018DA
system_type=IBM,9124-720
partition_id=3
R4=0xc8
R5=0x0
R6=0x80030000
R7=0x1000000030004
BoundThrds=1
CapInc=1
DisWheRotPer=1880000
MinEntCap=10
MinEntCapPerVP=10
MinMem=512
MinProcs=1
partition_max_entitled_capacity=200
system_potential_processors=4
DesEntCap=200
DesMem=4096
DesProcs=2
DesVarCapWt=0

partition_entitled_capacity=200
group=32771
system_active_processors=4
pool=0
pool_capacity=300
pool_idle_time=0
pool_num_procs=0
unallocated_capacity_weight=0
capacity_weight=0
capped=1
unallocated_capacity=0
purr=2862304508314
partition_active_processors=2
partition_potential_processors=4
```

```
shared_processor_mode=1
```

Where:

- serial_number: Provides the manufacturer and serial number.
- system_type: Provides Manufacturer, Type, and Model (this is an OpenPower 720).
- partition_id: Provides the LPAR ID number.
- CapInc: Provides a minimum CPU increment/decrement.
- MinEntCap: Provides a minimum entitled capacity for LPAR.
- MinEntCapPerVP: Provides a minimum entitled capacity required per virtual processor.
- MinMem: Provides a minimum required amount of main memory required for start of LPAR.
- MinProcs: Provides a minimum required number of virtual processors for the start of LPAR.
- partition_max_entitled_capacity: Provides a maximum entitled capacity for LPAR.
- system_potential_processors: Provides a maximum number of physical processors in the system.
- DesEntCap: Provides a desired entitled capacity for LPAR.
- DesMem: Provides a desired amount of main memory for LPAR.
- DesProcs: Provides a desired number of virtual processors for LPAR.
- DesVarCapWt: Provides a weight of LPAR when running in uncapped mode.
- system_active_processors: Provides the number of active physical processors in the system.
- pool_capacity:Provides a maximum capacity of the shared processor pool.
- capacity_weight: Provides the weight of LPAR when running in uncapped mode.
- capped: Determines whether LPAR runs in capped mode (0=FALSE, 1=TRUE).
- purr: Provides a calculation of consumed CPU cycles for an uncapped LPAR.
- partition_active_processors: Provides the number of active processors for this LPAR.
- partition_potential_processors: Provides the maximum number of possible virtual processors for LPAR.

- shared_processor_mode: Determines if the shared processor pool is being used (0=FALSE, 1=TRUE).

/proc/stat

Various pieces of information about kernel activity are available in the /proc/stat file. All of the numbers reported in this file are aggregates since the system first booted. The sample /proc/stat file on a LPAR configured with two processors is shown in the Example 5-30.

Example 5-30 Sample output to show /proc/stat

```
# cat /proc/stat
cpu  31910 353 8288 8759683 2166 5666 4714 8814
cpu0 15914 172 4284 1097553 418 3410 2216 3026
cpu1 15996 181 4003 7662129 1748 2256 2497 5787
intr 887540
ctxt 4149178
btime 1159202046
processes 5472
procs_running 1
procs_blocked 0
```

Where:

- cpu: The first line aggregates the numbers in all of the other “cpuN” lines.These numbers identify the amount of time the CPU has spent performing different kinds of work. Time units are in USER_HZ (typically hundredths of a second). The meanings of the columns are as follows, from left to right:
  - user: Normal processes executing in user mode
  - nice: Niced processes executing in user mode
  - system: Processes executing in kernel mode
  - idle: Self-explanatory
  - iowait: Waiting for I/O to complete
  - irq: Servicing interrupts
  - softirq: Servicing softirqs
- intr: This line gives counts of interrupts serviced since boot time, for each of the possible system interrupts. The first column is the total of all interrupts serviced; each subsequent column is the total for that particular interrupt.
- ctxt: This line gives the total number of context switches across all CPUs.
- btime: This line gives the time at which the system booted, in seconds since the UNIX epoch.

- processes: This line gives the number of processes and threads created, which includes (but is not limited to) those created by calls to the fork() and clone() system calls.
- procs_running: This line gives the number of processes currently running on CPUs.
- procs_blocked: This line gives the number of processes currently blocked and waiting for I/O to complete.

/proc/meminfo

Provides information about distribution and utilization of memory. The sample /proc/meminfo file on LPAR is shown in Example 5-31.

Example 5-31 Sample output to show /proc/meminfo

```
# cat /proc/meminfo
MemTotal:      4107448 kB
MemFree:       3697020 kB
Buffers:         48180 kB
Cached:         282836 kB
SwapCached:          0 kB
Active:         110944 kB
Inactive:       232040 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:      4107448 kB
LowFree:       3697020 kB
SwapTotal:     1526164 kB
SwapFree:      1526164 kB
Dirty:             140 kB
Writeback:           0 kB
Mapped:          23304 kB
Slab:            42888 kB
CommitLimit:   3579888 kB
Committed_AS:    49228 kB
PageTables:       1184 kB
VmallocTotal: 8589934592 kB
VmallocUsed:      2168 kB
VmallocChunk: 8589931584 kB
HugePages_Total:     0
HugePages_Free:      0
HugePages_Rsvd:      0
Hugepagesize:    16384 kB
```

The information comes in the form of both high-level and low-level statistics.

High-level statistics

- MemTotal: Total usable RAM (that is, physical RAM minus a few reserved bits and the kernel binary code).
- MemFree: Sum of LowFree+HighFree (overall stat).
- MemShared: 0; exists for compatibility reasons, but is always zero.
- Buffers: Memory in buffer cache. Mostly useless as a metric currently.
- Cached: Memory in the pagecache (diskcache) minus SwapCache.
- SwapCache: Memory that once was swapped out, is swapped back in, but is still in the swapfile (if memory is needed, it does not need to be swapped out AGAIN because it is already in the swapfile. This saves I/O.).

VM statistics

VM splits the cache pages into “active” and “inactive” memory. The idea is that if you need memory and some cache needs to be sacrificed for that, you take it from inactive memory, because you do not expect it to be used. The VM checks what is used on a regular basis and moves stuff around.

When you use memory, the CPU sets a bit in the pagetable and the VM checks that bit occasionally and based on that, it can move pages back to active memory. There is an order of “longest ago not used” in active memory (roughly; it is a little more complex in reality). The longest-ago used ones can get moved to inactive memory. Inactive memory is split into two party in kernel (2.4.18-24.8.0). Some have it in three parts.

- Active: Memory that has been used more recently and usually not reclaimed unless absolutely necessary.
- Inact_dirty: Dirty means “might need writing to disk or swap.” It takes more work to free. Some examples might be files that have not been written to yet. They are not written to memory too soon in order to keep the I/O down. For example, if you are writing logs, it might be better to wait until you have a complete log ready before sending it to disk.
- Inact_clean: Assumed to be easily freeable. The kernel will try to keep some clean memory around to have a bit of breathing room.
- Inact_target: A goal metric the kernel uses for making sure there are enough inactive pages around. When exceeded, the kernel will not do work to move pages from active to inactive. A page can also be made inactive in a few other ways. For example, if you do a long sequential I/O, the kernel assumes you are not going to use that memory and makes it inactive preventively. So you can get more inactive pages than the target because the kernel marks some cache as “more likely to be never used” and lets it cheat in the “last used” order.

Memory statistics

- HighTotal: The total amount of memory in the high region. Highmem is all memory above (approximately) 860 MB of physical RAM. The kernel uses indirect tricks to access the high memory region. The data cache can go in this memory region.
- LowTotal: The total amount of non-highmem memory.
- LowFree: The amount of free memory of the low memory region. This is the memory the kernel can address directly. All kernel datastructures need to go into low memory.
- SwapTotal: Total amount of physical swap memory.
- SwapFree: Total amount of swap memory free.
- Committed_AS: An estimate of how much RAM you would need to make a 99.99% guarantee that there never is out of memory (OOM) for this workload. Normally, the kernel will overcommit memory. So, if you do a 1 GB malloc, for example, nothing happens. Only when you start *USING* that malloc memory will you get real memory on demand and just as much as you use. As an analogy, it is like taking out a loan and hoping the bank does not go bust. Other cases might include when you map a file that is shared only when you write to it and you get a private copy of that data. While it normally is shared between processes, the Committed_AS is a guesstimate of how much RAM/swap you would need in the worst-case scenario.

5.6.4 sar

The **sar** command is used to collect, report, or save system activity information. The **sar** command writes to standard ouput the contents of selected cumulative activity counters in the operating system. The accounting system, based on the values in the count and interval parameters, writes information the specified number of times spaced at the specified interval in seconds. The output of the **sar** command to collect CPU utilization for 10 counts at an interval of one second on LPAR configured with two CPUs is shown in Example 5-32 on page 139.

Example 5-32 Ad hoc CPU monitoring

```
# sar -u 3 10
Linux 2.6.9-42.EL (rhel)  09/26/2006

05:43:53 PM       CPU     %user     %nice   %system   %iowait     %idle
05:43:56 PM       all      4.50      0.08     38.75      0.92     55.75
05:43:59 PM       all      5.08      0.17     40.50      0.08     54.17
05:44:02 PM       all      5.00      0.08     41.37      0.33     53.21
05:44:05 PM       all      5.07      0.17     41.18      0.50     53.08
05:44:08 PM       all      4.67      0.08     41.03      0.08     54.13
05:44:11 PM       all      4.50      0.08     39.53      0.58     55.30
05:44:14 PM       all      4.49      0.17     41.35      0.08     53.91
05:44:17 PM       all      4.66      0.08     40.55      0.67     54.04
05:44:20 PM       all      5.00      0.08     40.20      1.08     53.63

05:44:20 PM       CPU     %user     %nice   %system   %iowait     %idle
05:44:23 PM       all      4.91      0.08     41.38      0.50     53.12
Average:          all      4.79      0.11     40.58      0.48     54.03
```

The collected data shows a detailed overview of CPU utilization (%user, %nice, %system, %iowait, and %idle). Also, the average CPU utilization(%user, %nice, %system, %iowait, and %idele) is shown in the last line.

6

Identifying bottlenecks

A *bottleneck* is a congestion point in a system that occurs when workload increases beyond the point that the system can handle it. It results in lessening of throughput, because of the inability of the system to handle the workload. Often system analysts face such situations where the system performance deteriorates over a period of time. In such situations, performance bottlenecks are harder to identify because they can be related to hardware, firmware, operating system components, or even with the applications running or with combinations of the them. Without the set of rules or setups defined, it would be harder for the system analyst to identify the root cause for performance bottlenecks. This chapter discusses the sections in detail.

6.1 Steps to identify bottlenecks

The following steps are used as a quick tuning strategy:

- Know your system.
- Back up the system.
- Monitor and analyze the system performance.
- Root cause analysis of the bottleneck.

6.2 First information report

Most likely, the only first-hand information you will have access to will include statements such as “There is a problem with the server.” It is crucial to use probing questions to clarify, document, and prepare the first information report (FIR) of the problem. Here is a list of questions you should ask to help you get a better picture of the system and generate a FIR.

- Could you give me a complete description of the server in question?
  - Model.
  - Age.
  - Configuration.
  - Peripheral equipment.
  - Operating system version and update level.
  - Type of applications they run (such as WebSphere, Oracle, Apache, and DB2).
- Can you tell me what the problem is exactly?
  - What are the symptoms?
  - Description of any error messages.

  Some people will have problems answering this question. Any extra information the client can give you might help you discover the problem. For example, the client might say “It is really slow when I copy large files to the server.” This might indicate a network problem or a disk subsystem problem.
- Who is experiencing the problem?

  Is it one person, one particular group of people, or the entire organization experiencing the problem? This will help you determine whether the problem exists in one particular part of the network, whether it is application dependent, and so on. If only one user is experiencing the problem, then the problem might be with the user’s PC, or there may only be a perceived problem by the user.

The perception clients have of the server is usually a key factor. From this point of view, performance problems may not be directly related to the server: the network path between the server and the clients can easily be the cause of the problem. This path includes network devices as well as services provided by other servers, such as domain controllers.

- Can the problem be reproduced?

  All reproducible problems can be solved. If you have sufficient knowledge of the system, you should be able to narrow the problem to its root and decide which actions should be taken.

  The fact that the problem can be reproduced will allow you to see and understand it better. Document the sequence of actions necessary to reproduce the problem at any time:

  - What are the steps to reproduce the problem?

    Knowing the steps may let you reproduce the same problem on a different machine under the same conditions. If this works, it gives you the opportunity to use a machine in a test environment and removes the chance of crashing the production server.

  - Is it an intermittent problem?

    If the problem is intermittent, the first thing to do is to gather information and find a path to move the problem in the reproducible category. The goal here is to have a scenario to make the problem happen on command.

  - Does it occur at certain times of the day or certain days of the week?

    This might help you determine what is causing the problem. It may occur when everyone arrives for work or returns from lunch. Look for ways to change the timing (that is, make it happen less or more often); if there are ways to do so, the problem becomes a reproducible one.

  - Is it unusual?

    If the problem falls into the non-reproducible category, you may conclude that it is the result of extraordinary conditions and classify it as fixed. In real life, there is a high probability that it will happen again.

    A good procedure to troubleshoot a hard-to-reproduce problem is to perform general maintenance on the server: reboot, or bring the machine up-to-date on drivers and patches.

- When did the problem start? Was it gradual or did it occur very quickly?

  If the performance issue appeared gradually, then it is likely to be a sizing issue; if it appeared overnight, then the problem could be caused by a change made to the server or peripherals.

- Have any changes been made to the server (minor or major) or are there any changes in the way clients are using the server?

  Did the client alter something on the server or peripherals to cause the problem? Is there a log of all network changes available?

  Demands could change based on business changes, which could affect demands on servers and network systems.
- Are there any other servers or hardware components involved?
- Are there any logs available?
- What is the priority of the problem? When does it need to be fixed?
  - Does it need to be fixed in the next few minutes, or in days? You may have some time to fix it; or it may already be time to operate in panic mode.
  - How massive is the problem?
  - What is the related cost of that problem?

6.3 Monitoring system performance

System performance analysis require to monitor the system over a period of time and collect information to identify the bottlenecks. At this point, you should begin monitoring the server. The simplest way is to run monitoring tools from the server that is being analyzed.

Tip: Before you start monitoring the performance, it is always safe to back up all the data.

A performance log of the server should be created during its peak time of operation (for example, 9:00 a.m. to 5:00 p.m.); it will depend on what services are being provided and who is using these services. When creating the log, if available, the following objects should be included:

- Processor
- Memory
- System
- Physical disk
- Network interface

Before you begin, remember that a methodical approach to performance tuning is important. Our recommended process, which you can use for your System p server to identify performance bottlenecks, is as follows:

- Try to understand the logical connection of the various subsystems and the distinction between a hardware and a software bottleneck.
- Measure the current performance to create a performance baseline to compare with your future measurements and to identify system bottlenecks.
- Use the monitoring tools to identify a performance bottleneck. By following the instructions in the next sections, you should be able to narrow down the bottleneck to the subsystem level.

Many a times a hardware failure might be the cause of bottleneck. Before analyzing the software components, we advise making a list of the hardware initialized during booting maps to that of the present configuration. Use the `dmesg` tool to check for any hardware component failures.

Tip: Before analyzing software components, it is preferable to check the hardware components.

6.3.1 CPU performance indicators

Determining processor bottlenecks is easy. If the %Processor Time Total of any single processor is sustaining over 70-80% utilization, it should be considered a bottleneck. The reasons for the CPU bottlenecks can be:

- Most of the CPU time is allocated to run other unwanted processes and not the application it was intended to be running.
- The number of processes are so many that the CPU spends the context switching.

Table 6-1 shows the primary processor counters to determine if the CPU is a bottleneck using the commands discussed in 5.2, “CPU utilization monitoring tools” on page 105.

Table 6-1 CPU performance Indicators

Indicator	Analysis
Processor:% Processor Time	This is the primary counter for detecting processor bottlenecks. The CPU is a bottleneck if the value is consistently running over 70-80%.
Processor: % Privileged Time	This counter can be used to identify abnormally high kernel time and may indicate I/O driver problems.

Indicator	Analysis
Processor: % User Time	%User time represents the time spent by the user application on the server. This is an important counter because it shows a breakdown of how the server application is utilizing all the processors.
Processor: Interrupts/sec	Processor Interrupts/sec should no longer be used as a simple indicator of performance. Modern device drivers have dynamic mechanisms and batch interrupts when the system is busy, doing more work per interrupt. This causes the number of interrupts per second to be dynamic. When a system is moderately busy, it might require an interrupt to process each LAN or DISK I/O operation. In this mode, the server will have a fairly high interrupt rate. However, as the server becomes busier, multiple disk and LAN operations will be sent under one interrupt request, lowering the interrupt rate and improving interrupt efficiency.
Processor: % context switch	A normal acceptable range per process is 0- 25%. Use **`stap process.stp`** to collect this data. If the processor involves more context switches, it indicates a bottleneck.

An example of CPU bottlenecks are:

- A system running SUSE Linux Enterprise Server 10 with DB2 running on an LPAR configured with 1.2 CPU caused the bottleneck shown in Figure 6-1, where the average CPU utilization by an user application is 78% and the operating system is 30%. Many clients were simultaneously running applications that drove the workload on DB2. You can see that we had four virtual CPUs configured in this LPAR.

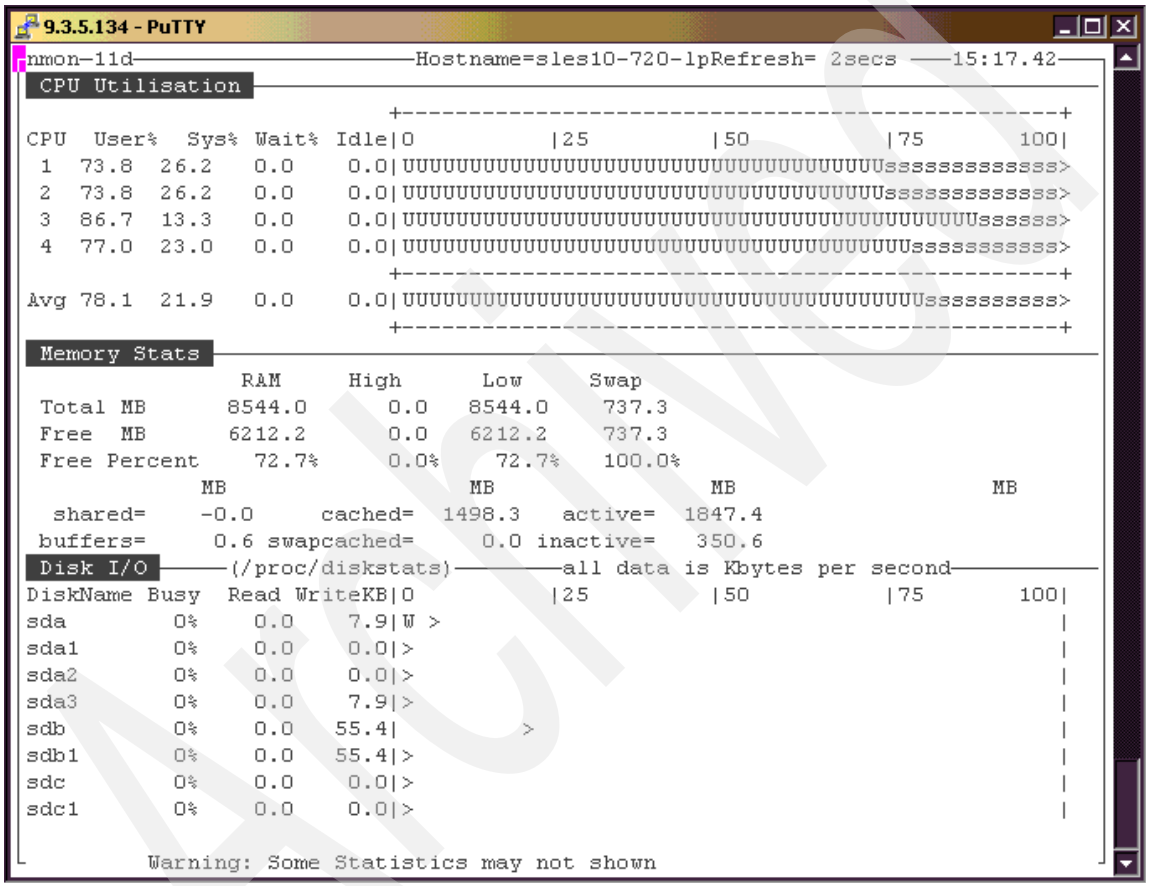


Figure 6-1 Showing CPU bottleneck on SUSE Linux Enterprise Server 10 running DB2

A system running Red Hat Enterprise Linux AS 4 update 4 with DB2 running on LPAR configured with 0.8 CPU caused the bottleneck shown in the Figure 6-2, where the average CPU utilization by user application is 58%, the operating system is 6%, and the wait is 35%. You can see on this LPAR that we have two virtual CPUs configured.

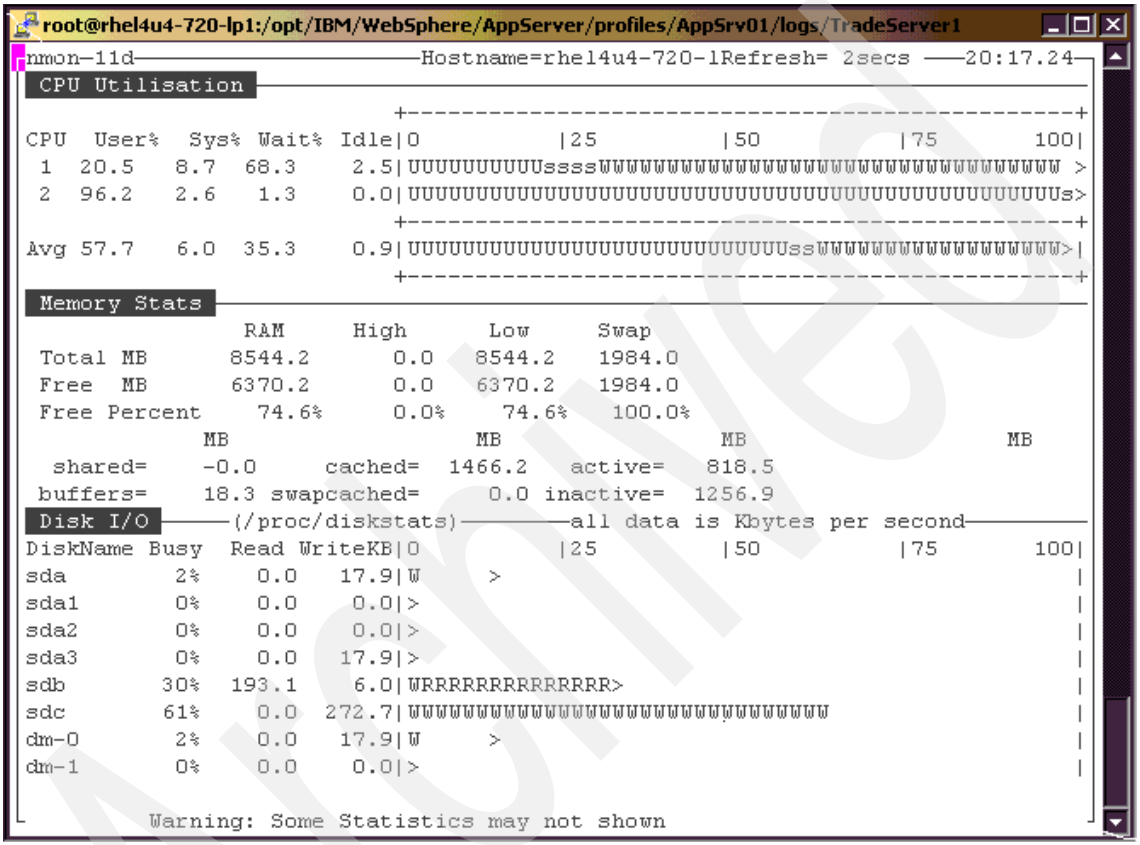


Figure 6-2 Showing CPU bottleneck on Red Hat Enterprise Linux AS 4, update 4 running DB2

6.3.2 Memory performance indicators

On a Linux system, many programs run at the same time; these programs support multiple users and some processes are used more than others. Some of these programs use a portion of memory while the rest are “sleeping”. When an application accesses a cache, the performance increases, because an in-memory access retrieves data, thereby eliminating the need to access slower disks.

The operating system uses an algorithm to control what programs will use physical memory, and which are paged out. This is transparent to user programs. Page space is a file created by the operating system on a disk partition to store user programs that are not currently being used. Typically, page sizes are 4 KB or 8 KB. In Linux, the page size is defined in the kernel header file include/asm-<architecture>/param.h using the variable EXEC_PAGESIZE. The process used to page a process out to disk is called *pageout*.

Start your analysis by listing the applications that are running on the server. Determine how much physical memory and swap each of the applications needs to run. The indicators that can help you determine the problem with the memory are shown in Table 6-2.

Table 6-2 Memory performance indicators

Memory indicator	Analysis
Memory available	This indicates how much physical memory is available for use. If, after you start your application, this value has decreased significantly, you may have a memory leak. Check the application causing it and make the necessary adjustments. For additional information, use: `#free -l -t -o` You can write the systemtap script to use the /usr/share/systemtap/tapset/memory.stp tapset to get the memory allocated and freed by using brk() system call for each application.
Page faults	There are two types of page faults: soft page faults, when the page is found in memory, and hard page faults, when the page is not found in memory and must be fetched from disk. Accessing the disk will slow your application down considerably. To get useful information for analyzing page faults, specifically columns pgpgin/s and pgpgout/s, use `#sar -B` You can write the systemtap script to use /usr/share/systemtap/tapset/memory.stp tapset to get the number of pagefaults per process basis, which will help you find the process or application causing the page faults.
Filesystem cache	This is the common memory space that a file system caches by using the command **`#free -l -t -o`** to process that attempt to write the shared pages that affect the file system cache.

Memory indicator	Analysis
Private memory for process	This represents the memory used by each process running on the server. You can see how much memory is allocated to a specific process using the **pmap** command. Also, you can write the systemtap script to use /usr/share/systemtap/tapset/memory.stp tapset to discover the process that causes do_mmap() and do_munmap(). You can use **pmap** to get per process mappings.
Page cluster	The page cluster controls the number of pages that are written to swap in a single attempt (the swap I/O size).The Linux VM subsystem avoids excessive disk seeks by reading multiple pages on a page fault. An improper configuration of a page cluster might cause server page faults.

6.3.3 Network performance indicators

A performance problem in the network subsystem can be the cause of many problems, such as a larger response time, system hang, unable to accept more connections, and packet loss.

It is important to remember that there are many possible reasons for these performance problems and that sometimes problems occur simultaneously, making it even more difficult to pinpoint the origin. The indicators that can help you determine the problem with your network are shown in Table 6-3.

Table 6-3 Network performance indicators

Network indicator	Analysis
Packets received Packets sent	Shows the number of packets that are coming in and going out of the specified network interface.
Collision packets	Collisions occur when there are many systems on the same domain. The use of a hub may be the cause of many collisions.
Dropped packets	Packets may be dropped for a variety of reasons, but the result may impact performance. For example, if the server network interface is configured to run at 100 Mbps full duplex, but the network switch is configured to run at 10 Mbps, the router may have an ACL filter that drops these packets, for example, **iptables -t filter -A FORWARD -p all -i eth2 -o eth1 -s 172.18.0.0/24 -j DROP**.
Errors	Errors occur if the communications lines (for example, the phone line) are of poor quality. In these situations, corrupted packets must be present, thereby decreasing network throughput.

Network indicator	Analysis
TCP connections	TCP is a connection oriented protocol. It is governed by the following parameters: ► tcp_max_syn_backlog ► tcp_synack_retries ► tcp_retries2 ► tcp_keepalive_time ► tcp_keepalive_intvl ► tcp_keepalive. If configured with an incorrect value, it might cause a performance bottleneck. Proper configuration of these value will result in better performance.
IP fragmentation	IP fragmentation is governed by the memory allocated for reassembly of IP fragments. ipfrag_high_thresh and ipfrag_low_thresh control the IP fragmentation memory allocation. Improver configured values might cause more fragmentation and slower network performance.

An example of network bottlenecks on SUSE Linux Enterprise Server 10 running on LPAR configured with two CPUs and a Gigabit Ethernet card is shown in Figure 6-3. The network bottleneck occurs while 800 concurrent clients tries to connect the Apache server for serving static content. Figure 6-3 shows that network bandwidth is 105 Mbps.

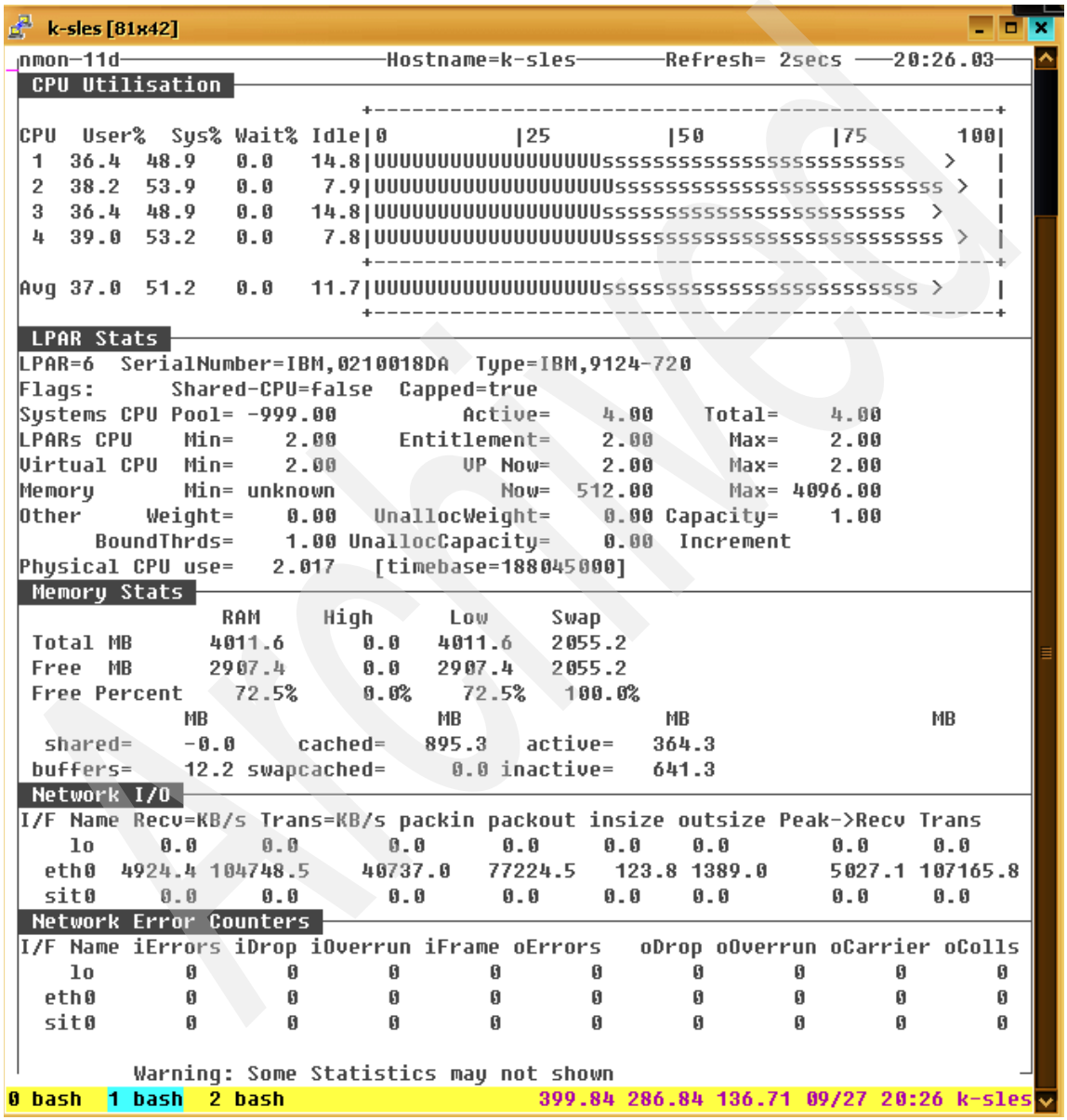


Figure 6-3 Shows potential network bottleneck

An example of potential network bottlenecks on SUSE Linux Enterprise Server 10 running on LPAR configured with two CPU with a Gigabit Ethernet card is shown in Example 6-1. There might be potential network bottleneck when 800 concurrent clients try to connect to the Apache server that serves static content. The out of the **netstat** command in Example 6-1 shows that there are 8391 passive TCP connection openings, 552 TCP connection attempts failed, and 8662 TCP connection received resets.

Example 6-1 Output of the netstat command to illustrate a potential bottleneck

```
Ip:
    28069393 total packets received
    9 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    28069384 incoming packets delivered
    53983067 requests sent out
Icmp:
    0 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
    0 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
Tcp:
    0 active connections openings
    48391 passive connection openings
    552 failed connection attempts
    8662 connection resets received
    796 connections established
    28068981 segments received
    53983069 segments send out
    25555 segments retransmitted
    0 bad segments received.
    157 resets sent
Udp:
    2 packets received
    0 packets to unknown port received.
    0 packet receive errors
    2 packets sent
TcpExt:
    98 resets received for embryonic SYN_RECV sockets
    ArpFilter: 0
    486742 delayed acks sent
    1126 delayed acks further delayed because of locked socket
```

```
Quick ack mode was activated 977 times
457 times the listen queue of a socket overflowed
457 SYNs to LISTEN sockets ignored
1 packets directly queued to recvmsg prequeue.
1 packets directly received from prequeue
1919444 packets header predicted
TCPPureAcks: 7823185
TCPHPAcks: 15811609
TCPRenoRecovery: 0
TCPSackRecovery: 1777
TCPSACKReneging: 0
TCPFACKReorder: 0
TCPSACKReorder: 0
TCPRenoReorder: 0
TCPTSReorder: 0
TCPFullUndo: 0
TCPPartialUndo: 0
TCPDSACKUndo: 0
TCPLossUndo: 64
TCPLoss: 1971
TCPLostRetransmit: 0
TCPRenoFailures: 0
TCPSackFailures: 506
TCPLossFailures: 9
TCPFastRetrans: 11936
TCPForwardRetrans: 1965
TCPSlowStartRetrans: 9661
TCPTimeouts: 1341
TCPRenoRecoveryFail: 0
TCPSackRecoveryFail: 130
TCPSchedulerFailed: 0
TCPRcvCollapsed: 0
TCPDSACKOldSent: 977
TCPDSACKOfoSent: 0
TCPDSACKRecv: 0
TCPDSACKOfoRecv: 0
TCPAbortOnSyn: 0
TCPAbortOnData: 25
TCPAbortOnClose: 503
TCPAbortOnMemory: 0
TCPAbortOnTimeout: 0
TCPAbortOnLinger: 0
TCPAbortFailed: 0
TCPMemoryPressures: 0
```

6.3.4 Disk I/O performance indicators

The disk subsystem is often the most important aspect of server performance and is usually the most common bottleneck. However, problems can be hidden by other factors, such as lack of memory. Applications are considered to be “I/O bound” when CPU cycles are wasted simply waiting for I/O tasks to finish.

The most common disk bottleneck is having too few disks. Most disk configurations are based on capacity requirements, not performance. The least expensive solution is to purchase the smallest number of the largest-capacity disks possible. However, this places more user data on each disk, causing greater I/O rates to the physical disk and allowing disk bottlenecks to occur.

The second most common problem is having too many logical disks on the same array. This increases seek time and greatly lowers performance.

The symptoms that show that the server may be suffering from a disk bottleneck (or a hidden memory problem) are shown in the Table 6-4.

Table 6-4 Disk I/O performance indicators.

Disk I/O indicators	Analysis
Disk I/O numbers and wait time	Analyze the number of I/Os to the individual disk. This data can be used to discover if reads or writes are the cause of problem. Use **`iostat`** to get the disk I/Os. Use **`stap ioblock.stp`** to get read/write blocks. Also, use **`scsi.stp`** to get the scsi wait times, requested submitted, and completed. Also, long wait times might mean the I/O is to specific disks and not spread out.
Disk I/O size	The memory buffer available for the block I/O request might not be sufficient and the page cache size can be smaller then the number of maximum number of Disk I/O size. Use **`stap ioblock.stp`** to get request sizes. Use **`iostat`** to get the block sizes.
Disk I/O scheduler	An incorrect I/O scheduler might cause performance bottlenecks. A certain I/O scheduler performs better if configured with the appropriate file system.

Disk I/O indicators	Analysis
Disk file system	An incorrect file system might cause performance bottlenecks. The appropriate file system must be chosen based on the requirements.
Disk I/O to physical device	If all the disk I/O are directed to the same physical disk, it might cause a disk I/O bottleneck. Directing the disk I/O to different physical disks will increase the performance.
File system block size	If the file system is created with small sized blocks, creating files larger than the block size might cause a performance bottleneck. Creating a file system with a proper block size would improve the performance.
Swap device/area	If a single swap device/area is used, then it might cause performance problems. To improve the performance, create multiple swap devices or areas.

Slow disks will result in memory buffers filling with write data (or waiting for read data), which will delay all requests, because free memory buffers are unavailable for write requests (or the response is waiting for read data in the disk queue), or insufficient memory, as in the case of not having enough memory buffers for network requests, will cause synchronous disk I/O.

Disk and controller utilization or both will typically be very high.

Most LAN transfers will happen only after disk I/O has completed, causing very long response times and low network utilization.

Since disk I/O can take a relatively long time, and disk queues will become full, the CPUs will be idle or have low utilization as they wait long periods of time before processing the next request.

Examples of disk I/O bottleneck are shown in Figure 6-4, where the LPAR configured with a 1.2 CPU running Red Hat Enterprise Linux AS 4 issues server writes to the disks sda and md2. Figure 6-4 also shows that the disk utilization is 100%.

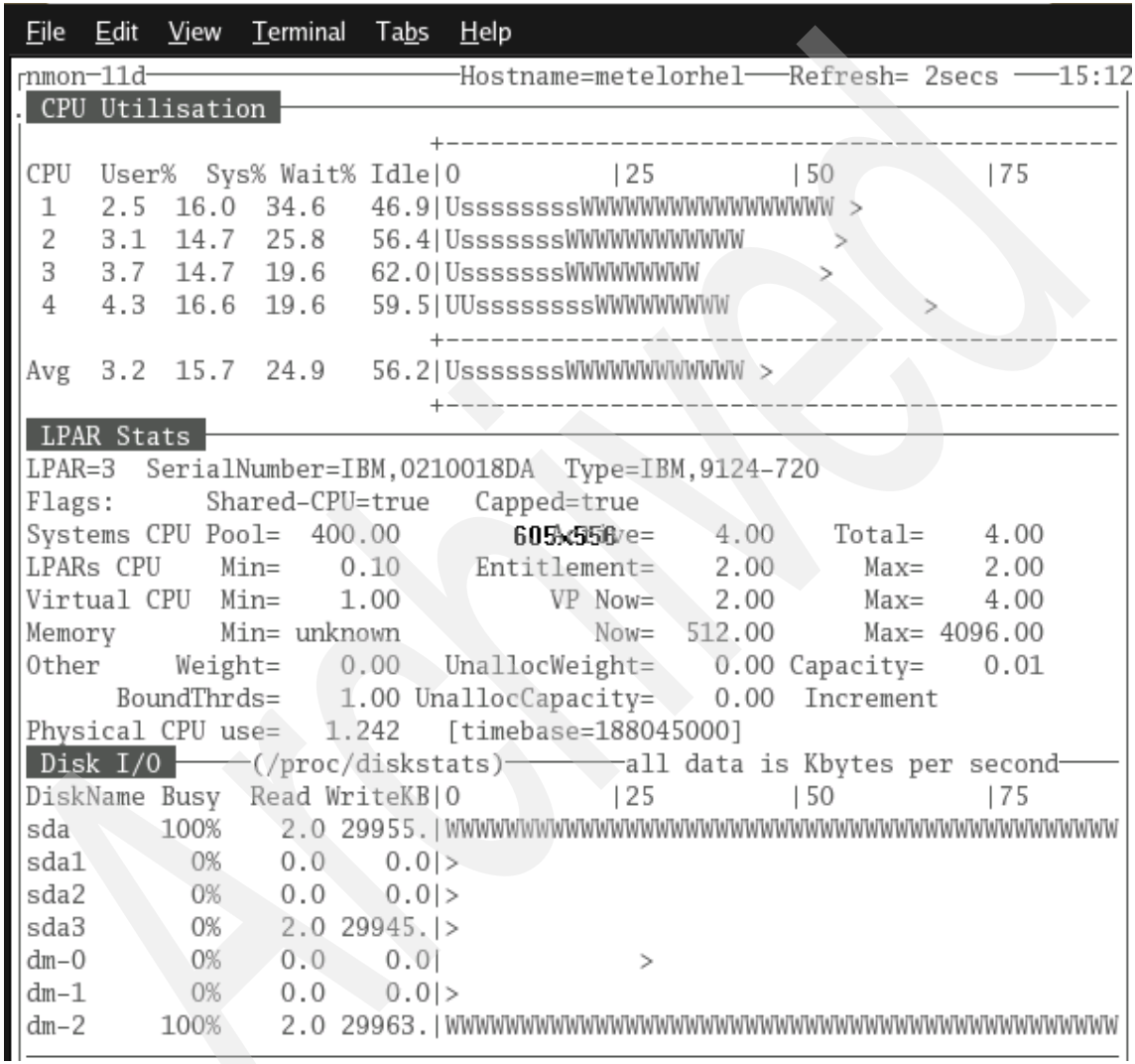


Figure 6-4 Shows the write I/O bottleneck

Example 6-2 shows the output of the **iostat** command on an LPAR configured with a 1.2 CPU running Red Hat Enterprise Linux AS 4 while issuing server writes to the disk sda and md2, where the transfers per second is 130 for sda and 692 for md-2. Also, the iowait is 6.35%.

Example 6-2 Shows output of iostat

```
#iostat
Linux 2.6.9-42.EL (rhel)  09/29/2006

avg-cpu:  %user   %nice    %sys %iowait   %idle
           2.70    0.11    6.50    6.37   84.32

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda             130.69      1732.56      5827.90     265688     893708
sda1              1.24         2.53         0.00        388          0
sda2              4.80         5.32         0.03        816          4
sda3            790.73      1717.40      5827.87     263364     893704
dm-0             96.19      1704.71       292.40     261418      44840
dm-1              0.29         2.35         0.00        360          0
dm-2            692.66         6.38      5535.47        978     848864
```

Example 6-3 shows the output of the **iostat** command on an LPAR configured with a 1.2 CPU running Red Hat Enterprise Linux AS 4 issuing server writes to the disk sda and md2, where the transfers per second is 428 for sda and 4024 for md-2. Also, the iowait has gone up to 12.42%.

Example 6-3 Shows output of iostat to illustrate disk I/O bottleneck

```
# iostat
Linux 2.6.9-42.EL (rhel)  09/29/2006
avg-cpu:  %user   %nice    %sys %iowait   %idle
           2.37    0.20   27.22   12.42   57.80

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda             428.14       235.64     32248.23     269840   36928420
sda1              0.17         0.34         0.00        388          0
sda2              0.64         0.71         0.00        816          4
sda3           4039.46       233.61     32248.17     267516   36928352
dm-0             14.63       231.80        52.47     265442      60080
dm-1              0.04         0.31         0.00        360          0
dm-2           4024.58         0.97     32195.76       1106   36868336
```

7

Linux kernel tuning

By its nature and heritage, the Linux distributions and the Linux kernel offer a variety of parameters and settings to let the Linux administrator tweak the system to maximize performance.

This chapter describes the steps you can take to tune Red Hat Enterprise Linux AS 4 Update 4 or SUSE Linux Enterprise Server10. The objective is to describe the parameters that give you the most improvement in performance. This chapter also gives you some basic understanding of the tuning techniques that are used in Linux.

This chapter includes the following sections:

7.1 Disabling daemons

There are daemons (background services) running on every server that are probably not needed. Disabling these daemons frees up memory, decreases startup time, and decreases the number of processes that the CPU has to handle. This will also provide increased security of the server, because fewer daemons mean fewer exploitable processes.

By default, several daemons are started that can be stopped safely on most servers. Table 7-1 applies to Red Hat Enterprise Linux AS 4 Update 4.

Table 7-1 Red Hat Enterprise Linux AS 4 U4: Tuning daemons started by default

Daemon	Description
autofs	Automatically mounts file systems on demand (that is, mounts a CD-ROM automatically)
cups	Common UNIX Printing System
hpoj	HP OfficeJet support
isdn	ISDN modem support
netfs	Used in support of exporting NFS shares
nfslock	Used for file locking with NFS
pcmcia	PCMCIA support on a server
portmap	Dynamic port assignment for RPC services (such as NIS and NFS)
rhnsd	Red Hat Network update service for checking for updates and security errata
sendmail	Mail Transport Agent
xfs	Font server for X Window System

Important: Turing off the xfs daemon will prevent the X Window System from starting on the server, so this should only be turned off if the server will not be booting into the GUI. Simply starting the xfs daemon before issuing the **`startx`** command will enable the X Window System to start normally.

Table 7-2 applies to SUSE Linux Enterprise Server 10.

Table 7-2 SUSE Linux Enterprise Server 10: Tunable daemons started by default

daemon	Description
alsasound	Sound daemon
portmap	Dynamic port assignment for RPC services (such as NIS and NFS)
postfix	Mail Transport Agent
splash	Splash window setup
fbset	Framebuffer setup
splash_early	Kills animation after the network starts
xdm	X Window System display manager
cups	Common UNIX Printing System

You can stop a daemon immediately if required. For example, the sendmail daemon can be stopped using the following commands:

- Red Hat Enterprise Linux AS 4 U4

  ```
  /sbin/service sendmail stop
  ```

- SUSE Linux Enterprise Server 10

  ```
  /etc/init.d/sendmail stop
  ```

You can also configure the daemon to not start the next time the server starts. Again, for the sendmail daemon, you can use the following commands:

- Red Hat Enterprise Linux AS 4 U4

  ```
  /sbin/chkconfig sendmail off
  ```

- SUSE Linux Enterprise Server 10

  ```
  /sbin/chkconfig -s sendmail off
  ```

In addition, these distributions provide a graphical interface to managing daemons.

For Red Hat Enterprise Linux AS 4 Update 4, to run the GUI, issue the command:

```
/usr/bin/system-config-services
```

or

Select **Main Menu** → **System Settings** → **Server Settings** → **Services**. Figure 7-1 shows Red Hat Enterprise Linux AS4 Configuration GUI.

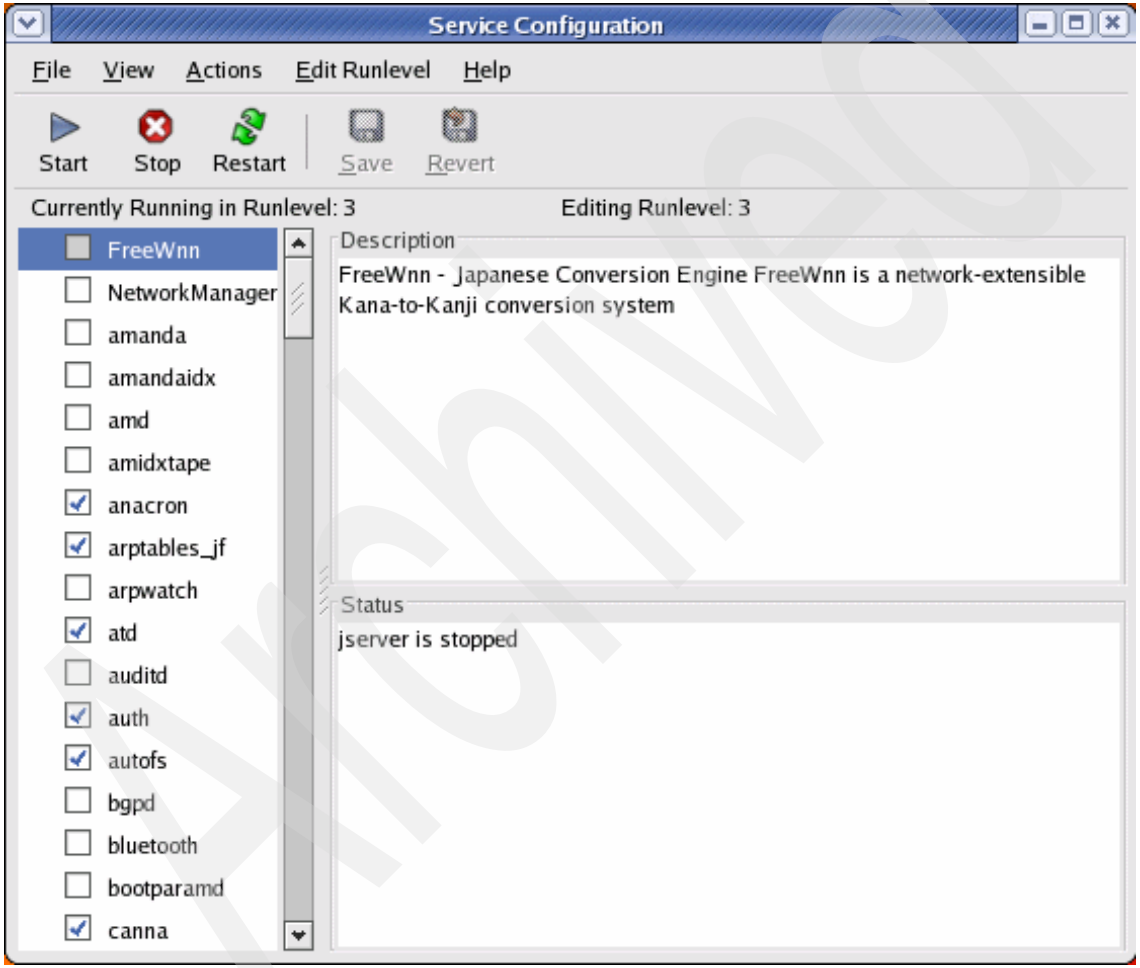


Figure 7-1 Shows Red Hat Enterprise Linux AS4 Configuration GUI

Tip: Not all daemons appear in the Red Hat Enterprise Linux AS 4 Update 4 Configuration window. To see a complete list, issue the command:

```
/sbin/chkconfig --list
```

For SUSE Linux Enterprise Server 10, the graphical interface is YaST2, which can be started either with the command **`/sbin/yast2 runlevel`** or by selecting **Browse** → **YaST/** → **YaST modules** → **System** → **Runlevel editor**, as shown in Figure 7-2.

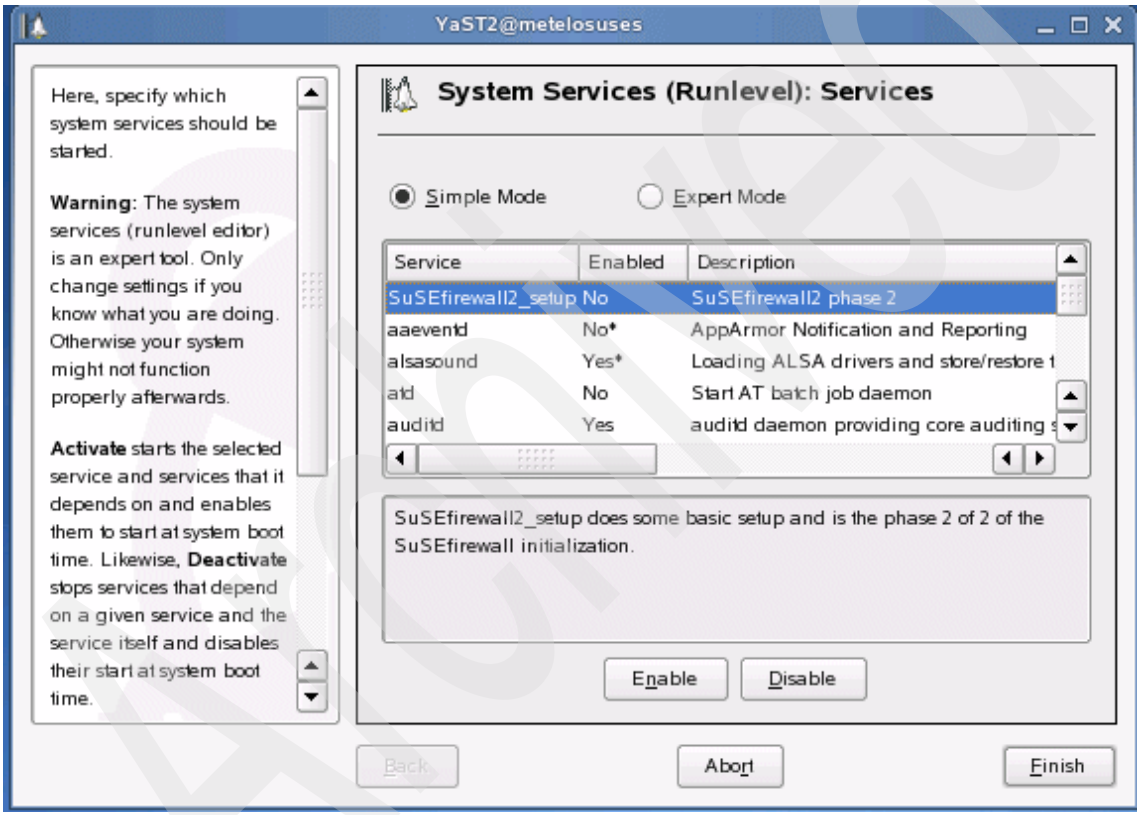


Figure 7-2 Shows SUSE Linux Enterprise Server 10 YaST2 Runlevel editor

7.2 Shutting down the GUI

Whenever possible, do not run the GUI on a Linux server. Normally, there is no need for a GUI on a Linux server. All administration tasks can be performed through the command line, by redirecting the X Window System display or through a Web browser interface. There are several different useful Web-based tools (for example, webmin, Linuxconf, and SWAT).

If there is really a need for a GUI, then start and stop it as needed rather than running it all the time. In most cases, the server should be running at runlevel 3, which does not start the GUI when the machine boots. If you want to restart the X Window System server, use **`startx`** at a command prompt.

- Determine which runlevel the machine is running by using the command:

  ```
  runlevel
  ```

  This prints the previous and current runlevel (for example, N 5 means that there was no previous runlevel (N) and that the current runlevel is 5).

- To switch between runlevels, use the **`init`** command. For example, to switch to run level 3, enter the command:

  ```
  init 3
  ```

Here is a short description of the different runlevels that are used in Linux:

- 0 - Halt (Do not set initdefault to this or the server will immediately shut down after finishing the boot process.)
- 1 - Single user mode
- 2 - Multi-user, without NFS (the same as 3, if you do not have networking)
- 3 - Full multi-user mode
- 4 - Unused
- 5 - X11
- 6 - Reboot. (Do not set initdefault to this or the server machine will continuously reboot at startup.)

To set the initial runlevel of a machine at boot time, modify the /etc/inittab file, as shown in Figure 7-3. With SUSE Linux Enterprise Server 10, this can also be accomplished by running the **YaST** runlevel command and changing the default runlevel.

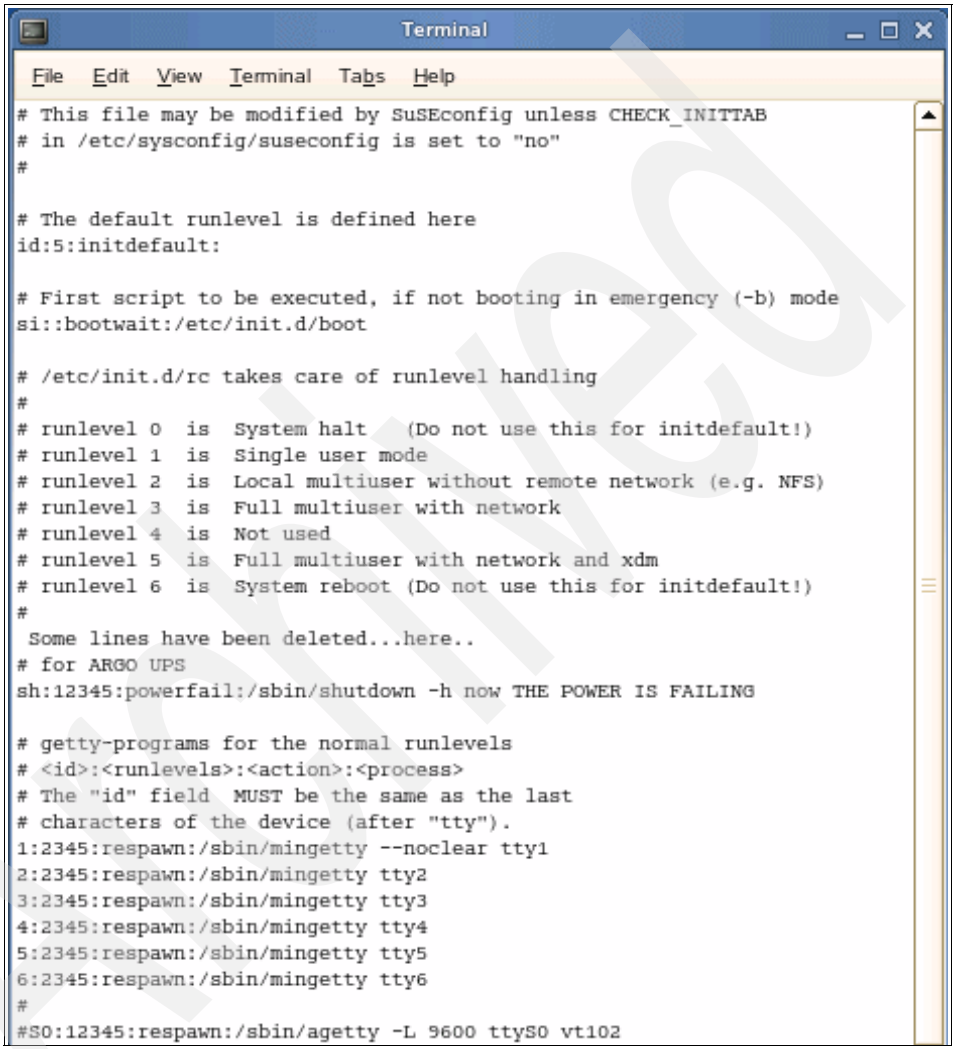


Figure 7-3 Showing parts of /etc/inittab file

By default, six consoles are saved: F1 through F6 are separate consoles. To regain some memory, you may wish to limit the number of consoles to three from the original six. To do this, comment out each **mingetty ttyx** line you want to disable.

7.3 Changing kernel parameters

The Linux kernel is the core of the operating system (OS) and is common to all Linux distributions. You can make changes to the kernel by modifying parameters that control the OS. These changes are made on the command line using the **`sysctl`** command.

Tip: By default, the kernel includes the necessary module to enable you to make changes using **`sysctl`** without needing to reboot. However, If you choose to remove this support (during the operating system installation), then you will have to reboot Linux before the change can take effect.

SUSE Linux Enterprise Server 10 offers a graphical method of modifying these sysctl parameters. To launch the powertweak tool, issue the following command:

```
/sbin/yast powertweak
```

or by select **Browse** → **YaST/** → **YaST modules** → **System** → **powertweak**.

Figure 7-4 shows the SUSE Linux Enterprise Server 10 powertweak tool.

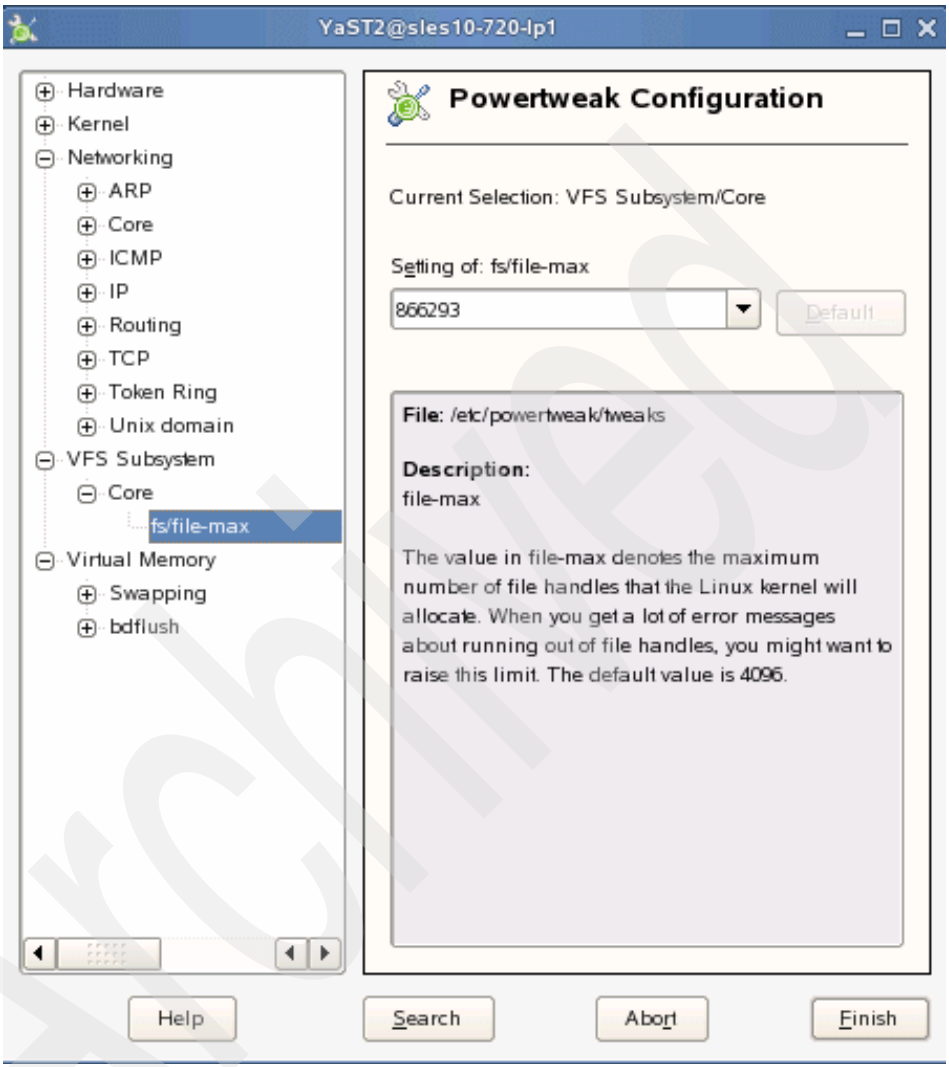


Figure 7-4 SUSE Linux Enterprise Server 10 powertweak tool

Red Hat Enterprise Linux AS 4 - Update 4 does not offer a graphical tool to modify the kernel parameter. Kernel parameters can be modified using the **sysctl** command:

```
#sysctl <parameter storage location>
```

For details about the usage of **sysctl**, refer to 7.3.1, “Using the sysctl commands” on page 169. The kernel parameters that control how the kernel behaves are stored in /proc (and in particular, /proc/sys).

Reading the files in the /proc directory tree provides a simple way to view configuration parameters that are related to the kernel, processes, memory, network, and other components. Each process running in the system has a directory in /proc with the process ID (PID) as name. Table 7-3 lists some of the files that contain kernel information.

Table 7-3 Showing /proc directory tree

File/directory	Purpose
/proc/loadavg	Information about the load of the server in 1-minute, 5-minute, and 15-minute intervals. The **uptime** command gets information from this file.
/proc/kcore	(SUSE Linux Enterprise Server 10 only) Contains data to generate a core dump at , for kernel debugging purposes. The command to create the core dump is **gdb**: `#gdb /usr/src/linux/vmlinux /proc/kcore`
/proc/stat	Kernel statistics, such as process, swap, and disk I/O.
/proc/cpuinfo	Information about the installed CPUs.
/proc/meminfo	Information about memory usage. The **free** command uses this information.
/proc/sys/fs/*	Used to increase the number of open files the OS allows and to handle quota.
/proc/sys/kernel/*	For tuning purposes, you can enable hotplug, manipulate shared memory, and specify the maximum number of pid files and level of debug in syslog.
/proc/sys/net/*	Tuning of network in general, IPV4 and IPV6.
/proc/sys/vm/*	Management of cache memory and buffer.

7.3.1 Using the sysctl commands

The **sysctl** commands use the names of files in the /proc/sys directory tree as parameters. Example 7-1 shows how to modify the shmmax kernel parameter so you can display (using **cat**) and change (using **echo**) the file /proc/sys/kernel/shmmax.

Example 7-1 Viewing and modifying the shmmax parameter

```
#cat /proc/sys/kernel/shmmax
33554432
#echo 33554430 > /proc/sys/kernel/shmmax
#cat /proc/sys/kernel/shmmax
33554430
```

However, using these commands can easily introduce errors, so we recommend that you use the **sysctl** command, because it checks the consistency of the data before it makes any change, as shown in Example 7-2.

Example 7-2 Viewing and modifying the shmmax parameter

```
#sysctl kernel.shmmax
kernel.shmmax = 33554432
#sysctl -w kernel.shmmax=33554430
kernel.shmmax = 33554430
#sysctl kernel.shmmax
kernel.shmmax = 33554430
```

This change to the kernel will stay in effect only until the next reboot. If you want to make the change permanent, you can edit the /etc/sysctl.conf or /etc/sysconfig/sysctl file and add the appropriate command. In our example:

```
kernel.shmmax = 33554439
```

The next time you reboot, the parameter file will be read. You can do the same thing with out rebooting by issuing the following command:

```
#sysctl -p
```

7.4 Kernel parameters

The Linux kernel has many parameters that can improve performance for your installation.

Table 7-4 shows the kernel parameters that are common and most relevant to performance for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 Update 4.

Table 7-4 Common tunable kernel parameters

Parameter	Description / example of usage
kernel.shmmax	Normally used for tuning database servers. This value can be used to query and set the limit on the maximum shared memory segment size that can be created. Shared memory segments up to 1 GB are now supported in the kernel. For Red Hat Enterprise Linux AS 4U4, the default value is 32 kilobytes. For example: `sysctl -w kernel.shmmax=0x8000000`
kernel.msgmax	This variable is used to specify the maximum message size in bytes. The default value for SUSE Linux Enterprise Server 10 is 64 KB and for Red Hat Enterprise Linux AS 4 U4 is 8 KB.
kernel.msgmnb	This variable specifies the default maximum size of the message queue. The default value for SUSE Linux Enterprise Server 10 is 64 KB and for Red Hat Enterprise Linux AS 4 U4 is 16 KB.
kernel.msgmni	This variable specifies the maximum number of message queue identifiers. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 16.
kernel.shmmni	This variable specifies the maximum number of shared memory segments that can be created systemwide. The default is for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 4096. It can be increased based on usage.

Parameter	Description / example of usage
kernel.shmall	This variable specifies the maximum shared memory segments system wide in pages. the default value on Red Hat Enterprise Linux AS 4 U4 is 2 MB. It can be increased if the usage of shared memory system wide increases.
net.ipv4.conf.all.arp_announce	Define different restriction levels for announcing the localsource IP address from IP packets in ARP requests sent on the interface: 0 - (default) Use any local address, configured on any interface. 1 - Try to avoid local addresses that are not in the target's subnet for this interface. This mode is useful when target hosts reachable through this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request, we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet, we select source address according to the rules for level 2. 2 - Always use the best local address for this target. In this mode, we ignore the source address in the IP packet and try to select the local address that we prefer for talks with the target host. Such a local address is selected by looking for primary IP addresses on all our subnets on the outgoing interface that include the target IP address. The max value from conf/{all,interface}/arp_announce is used. Increasing the restriction level gives more chance for receiving an answer from the resolved target while decreasing the level announces more valid sender's information. For example: `sysctl -w net.ipv4.conf.all.arp_annonce=2`

Parameter	Description / example of usage
net.ipv4.conf.default.arp_announce	Define different restriction levels for announcing the localsource IP address from IP packets in ARP requests sent on interface. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 0. For example: `sysctl -w net.ipv4.conf.default.arp_annonce=2`
net.ipv4.conf.eth0.arp_announce	Define the different restriction levels for announcing the localsource IP address from IP packets in ARP requests sent on the interface. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 0. For example: `sysctl -w net.ipv4.conf.eth0.arp_announce=1`
net.ipv6.conf.all.mtu	The default maximum for transfer unit on IPV6. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 1280. For example: `sysctl -w net.ipv6.conf.all.mtu=9000`
net.ipv6.conf.all. router_solicitation_delay	Determines whether to wait after the interface opens before sending router solicitations. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 1. (The kernel should wait.) For example: `sysctl -w net.ipv6.conf.all.router_solicitation_delay=0`
net.ipv6.conf.all. router_solicitation_interval	The number of seconds to wait between router solicitations. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is four seconds. For example: `sysctl -w net.ipv6.conf.all.router_solicitation_interval=3`

Parameter	Description / example of usage
net.ipv6.conf.all. router_solicitations	The number of router solicitations to send until assuming no routers are present. The default value is 3 for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4. For example: `sysctl -w net.ipv6.conf.all.router_solicitations=2`
net.ipv6.conf.all.temp_prefered_lft	The lifetime preferred, in seconds, for temporary addresses. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 86400 (1 day). For example: `sysctl -w net.ipv6.conf.all.temp_prefered_lft=259200`
net.ipv6.conf.all.temp_valid_lft	The lifetime, valid in seconds, for a temporary address. The default for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 604800 (one week). For example: `sysctl -w net.ipv6.conf.all.temp_valid_lft=302400`
net.ipv6.conf.default. accept_redirects	Accepts redirects sent by a router that works with IPV6, but it cannot set if forwarding is set to enable. Always one or other, they can never set together, because it will cause problems in all-IPV6 networks. The default value for Red Hat Enterprise Linux AS 4 and SUSE Linux Enterprise Server 10 is 1 (enabled). For example: `sysctl -w net.ipv6.conf.default.accept_redirects=0`

Parameter	Description / example of usage
net.ipv4. inet_peer_gc_maxtime	How often the garbage collector (gc) should pass over the inet peer storage memory pool during low or absent memory pressure. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 120, measured in jiffies. For example: `sysctl -w net.ipv4.inet_peer_gc_maxtime=240`
net.ipv4. inet_peer_gc_mintime	Sets the minimum time that the garbage collector can pass cleaning memory. If your server is heavily loaded, you may want to increase this value. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 10, measured in jiffies. For example: `sysctl -w net.ipv4.inet_peer_gc_mintime=80`
net.ipv4.inet_peer_maxttl	The maximum time-to-live for the inet peer entries. New entries will expire after this period of time. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 600, measured in jiffies. For example: `sysctl -w net.ipv4.inet_peer_maxttl=500`
net.ipv4. inet_peer_threshold	Set the size of inet peer storage. When this limit is reached, peer entries will be thrown away, using the inet_peer_gc_mintime timeout. The default value for SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4 U4 is 65644. For example: `sysctl -w net.ipv4.inet_peer_threshold=65644`

7.5 Tuning processor subsystem

The CPU is one of the most important hardware subsystems for servers whose primary role is that of an application or database server. However, in these systems, the CPU is often the source of performance bottlenecks.

For information about the tweaking of processor tuning parameters, refer to 7.4, “Kernel parameters” on page 170.

7.5.1 Enabling and disabling SMT

As a requirement for performance improvements at the application level, simultaneous multi-threading functionality is embedded in the POWER5 chip technology. Applications developed to use process-level parallelism (multi-tasking) and thread-level parallelism (multi-threads) can shorten their overall execution time. Simultaneous multi-threading is the next stage of processor saturation for throughput-oriented applications to introduce the method of instruction-level parallelism to support multiple pipelines to the processor.

The simultaneous multi-threading mode maximizes the usage of the execution units. In the POWER5 chip, more rename registers have been introduced (for floating-point operation, rename registers increased to 120), which are essential for out-of-order execution, and then vital for simultaneous multi-threading.

If simultaneous multi-threading is activated:

- More instructions can be executed at the same time.
- The operating system views twice the number of physical processors installed in the system.
- Provides support in mixed environments
  - Capped and uncapped partitions
  - Virtual partitions
  - Dedicated partitions
  - Single partition systems

Note: Simultaneous multi-threading is supported on POWER5 processor-based systems running Linux operating system-based systems at an appropriate level.

The simultaneous multi-threading policy is controlled by the operating system and is thus partition specific.

On Linux operating systems, you can disable or enable SMT only by using a Linux boot console command and a system reboot. You need to append the following command line in the /etc/yaboot.conf file:

```
append=”smt-enabled=off"
```

7.5.2 Allocating more CPUs

To increase the processing units or processor entitlement for a particular partition, you can change the CPU resources. You can make a change to the number of virtual processors in a shared processor partition using Dynamic LPAR (DLPAR) functions. Using the HMC or VIO, you can change the resources of a Dynamic LPAR. Here are the steps to perform on the HMC:

1. Open **Server and Partition** and select **Server Management**.
2. Open the server that contains the partition, and open the partitions.
3. Right-click the logical partition and select **Dynamic Logical Partitioning**.
4. Select the resource type that should be changed: **Processor Resources**.
5. Select **Add**, **Remove**, or **Move** as shown in Figure 7-5 on page 177.

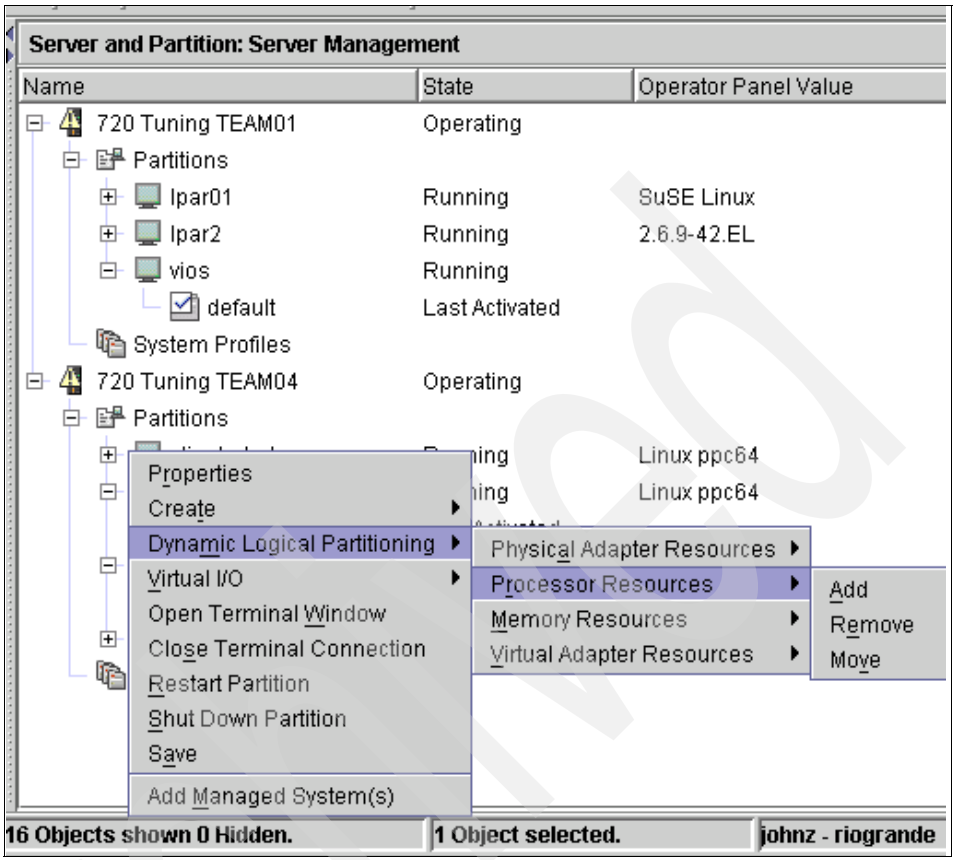


Figure 7-5 Shows dialog for changing number of virtual processor

6. When you select **Add**, you will see the dialog box shown in Figure 7-6.

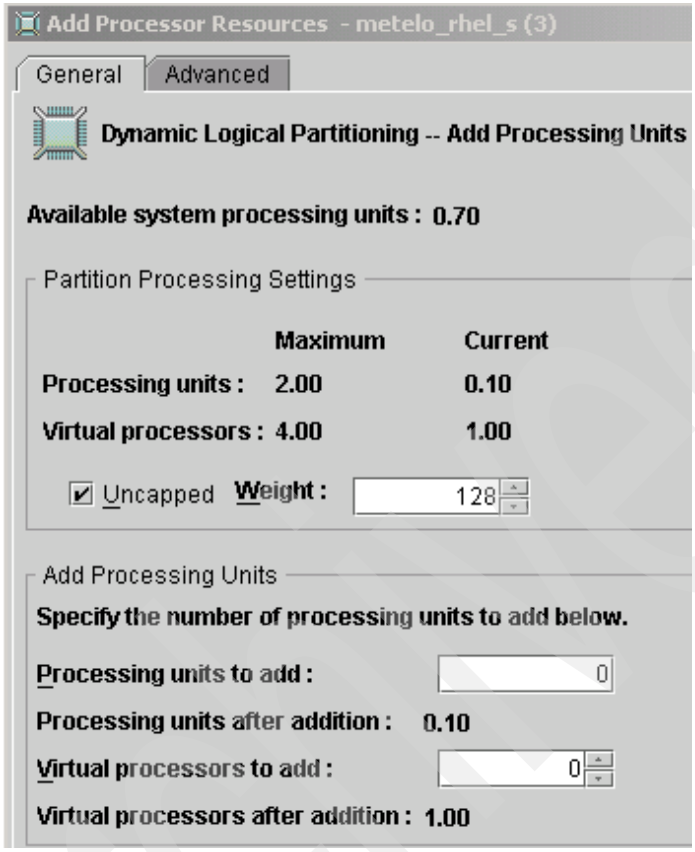


Figure 7-6 Shows dialog to add virtual and physical processor

After using the HMC to make dynamic LPAR changes, you can verify the partition resource changes to see the effect of the changes. The content of the /proc/ppc64/lparcfg file provides the current status of the LPAR's processor resources. Figure 7-7 on page 179 shows the lparcfg file for the partition after adding the virtual processor. The number of virtual processors, or a dedicated partition's number of processors, can be seen in partition_active_processors. The value of a shared processor partition's processing units can be seen in partiition_entitled_capacity. Also notice shared_processor_mode =1, since this a shared processor partition.

Figure 7-7 Shows /proc/ppc64/lparcfg after adding a virtual processor

7.6 Tuning Memory subsystem

A Linux virtual memory subsystem provides various features, such as highmemory, pagecache, page swapping, writeback to the disk, available free memory, page clustering, and so on. Tuning the memory subsystem is a difficult task that requires constant monitoring to ensure that changes do not negatively affect other subsystems in the server. If you do choose to modify the virtual memory parameters (in /proc/sys/vm), we recommend that you change only one parameter at a time, as shown in the Example 7-3, and monitor how the server performs.

Example 7-3 Shows the usage of sysctl to change the default parameter

```
#sysctl vm.page-cluster
vm.page-cluster = 3
#sysctl -w vm.page-cluster=1
This command change the default value os page-cluster from 3 to 1.
which means 2 pages are written to swap in a single attempt.
```

Table 7-5 shows the virtual memory tunable parameters.

Table 7-5 Virtual Memory Tunable parameter

Tunable parameter	Analysis
dirty_background_ratio	Contains, as a percentage of total system memory, the number of pages at which the pdflush background writeback daemon will start writing out dirty data.
dirty_ratio	Contains, as a percentage of total system memory, the number of pages at which a process that is generating disk writes will itself start writing out dirty data.
dirty_writeback_centisecs	The pdflush writeback daemon will periodically wake up and write old data out to disk. This tunable expresses the interval between those wakeups, in hundredths of a second.
dirty_expire_centisecs	This tunable is used to define when dirty data is old enough to be eligible for writeout by the pdflush daemons. It is expressed in hundredths of a second. Data that has been dirty in-memory for longer than this interval will be written out next time a pdflush daemon wakes up.

Tunable parameter	Analysis
legacy_va_layout	If non-zero, this `sysctl` disables the new 32-bit mmap mmap layout; the kernel will use the 2.4 layout for all processes
lower_zone_protection	For some specialized workloads on highmem machines, it is dangerous for the kernel to allow process memory to be allocated from the "lowmem" zone. Setting lower_zone_protection=100 will protect around 100 MB of the lowmem zone from user allocations. It will also make those 100 MB unavailable for use by applications and by pagecache, so there is a cost. A reasonable value for lower_zone_protection is 100.
page-cluster	page-cluster controls the number of pages that are written to swap in a single attempt. (the swap I/O size.) The Linux VM subsystem avoids excessive disk seeks by reading multiple pages in a page fault. The number of pages it reads is dependent on the amount of memory in your machine. It is a logarithmic value; setting it to zero means one page, setting it to 1 means two pages, setting it to 2 means four pages, and so on. The default value is three (eight pages at a time). There may be some small benefits in tuning this to a different value if your workload is swap-intensive.

Tunable parameter	Analysis
overcommit_memory	Controls overcommit of system memory, possibly allowing processes to allocate (but not use) more memory than is actually available. ► 0 Heuristic overcommit handling. Obvious overcommits of address space are refused. Used for a typical system. It ensures a seriously wild allocation fails while allowing overcommit to reduce swap usage. The root user is allowed to allocate slightly more memory in this mode. This is the default. ► 1 Always overcommit. Appropriate for some scientific applications. ► 2 Do not overcommit. The total address space commit for the system is not permitted to exceed swap plus a configurable percentage (default is 50) of physical RAM. Depending on the percentage you use, in most situations this means a process will not be killed while attempting to use already-allocated memory, but will receive errors on memory allocation as appropriate.
overcommit_ratio	This parameter specifies the percentage of physical memory that is considered when the overcommit_memory parameter is set to 2. The default value is set to 50.
nr_hugepages	Configures the number of hugetlb pages reserved for the system.
hugetlb_shm_group	Contains the group ID that is allowed to create SysV shared memory segment using hugetlb page.

Tunable parameter	Analysis
max_map_count	This file contains the maximum number of memory map areas a process may have. Memory map areas are used as a side-effect of calling malloc, directly by mmap and mprotect, and also when loading shared libraries. While most applications need less than a thousand maps, certain programs, particularly malloc debuggers, may consume many of them, for example, up to one or two maps per allocation. The default value is 65536.
min_free_kbytes	This is used to force the Linux VM to keep a minimum number of kilobytes free. The VM uses this number to compute a pages_min value for each lowmem zone in the system. Each lowmem zone gets a number of reserved free pages based proportionally on its size.
percpu_pagelist_fraction	This is the fraction of pages at most (high mark pcp $\rightarrow$ high) in each zone that are allocated for each per CPU page list. The min value for this is 8. It means that we do not allow more than one-eight of the pages in each zone to be allocated in any single per_cpu_pagelist. This entry only changes the value of hot per CPU pagelists. The user can specify a number like 100 to allocate one-hundredth of each zone to each per CPU page list. The batch value of each per CPU pagelist is also updated as a result. It is set to pcp->high/4. The upper limit of batch is (PAGE_SHIFT * 8). The initial value is zero. The kernel does not use this value at boot time to set the high water marks for each per CPU page list.

Tunable parameter	Analysis
nr_pdflush_threads	Governs the current size of the pdflush (write-back) thread pool. The VM subsystem utilizes a dynamic algorithm to determine the number of actual threads. The focus is on reserving a certain number of pdflush threads in case of a low-memory situation. The lower and upper bounds are 2 and 8.
swappiness	If the system encounters a situation where it is difficult to locate unmapped pages that can be reclaimed, the VM subsystem reverses its course and starts paging out anonymous memory. The preference of paging out anonymous memory can be expressed by specifying a lower value for swappiness. If kswapd is utilizing significant CPU resources or the sytem is spending time in an iowait state, increasing the value for swappiness may increase overall system performance.

7.7 Tuning Disk I/O subsystem

One of the most important features of Linux is that it supports many different file systems. This makes it very flexible and many file systems can coexist with many other operating systems. Linux distributions, such as Red Hat Enterprise Linux AS4 Update 4 and SUSE Linux Enterprise Server 10, support various file systems, such as ext2, ext3, xfs, msdos, vfat, proc, smb, iso9660, hpfs, affs, ufs, tmpfs, sysfs, resierfs, and so on. Red Hat Enterprise Linux AS 4 Update 4 supports ext3 as the default file system and does not support xfs. SUSE Linux Enterprise Server 10 supports reiserfs as the default file sytem.

The files in a file system are collections of data. A file system not only holds the data that is contained within the files of the file system, but also the structure of the file system. It holds all of the information that Linux users and processes see as files, directories soft links, file protection information, and so on. Moreover, it must hold that information safely and securely, as the basic integrity of the operating system depends on its file systems.

Linux adds each new file system into a single file system tree as it is mounted. All file systems, of whatever type, are mounted onto a directory and the files of the mounted file system cover up the existing contents of that directory. This directory is known as the *mount directory* or *mount point*. When the file system is unmounted, the mount directory's own files are once again revealed. When disks are initialized (using fdisk, for example) they have a partition structure imposed on them that divides the physical disk into a number of logical partitions. Each partition may hold a single file system, for example, an ext2 file system. 7.7.2, “Tuning file systems” on page 192 section discusses the ext3, xfs, and reiserfs file systems.

Capacity requirements are often the only consideration that is used to determine the number of disk drives that are configured in a server. Throughput requirements are usually not well understood or are completely ignored. Good performance by the disk subsystem depends on maximizing the number of read-write heads that can service I/O requests.

With redundant array of independent disks (RAID) technology, you can spread the I/O over multiple spindles. There are two options for implementing RAID in a Linux environment: software RAID or hardware RAID. Many System p servers ship with hardware RAID support, but if not, you may want to start with the software RAID options that come with the Linux distributions.

Software RAID in the 2.6 Linux kernel distributions is implemented through the md device driver. This driver implementation is device-independent and therefore is flexible in allowing many types of disk storage, such as EIDE or SCSI, to be configured as a RAID array. Supported software RAID levels are RAID 0 (striping), RAID 1 (mirroring), and RAID 5 (striping with parity), and can be implemented as part of the initial installation or through the mdadm tool set.

7.7.1 Tuning the disk I/O scheduler

The I/O scheduler forms the interface between the generic block layer and the low-level device drivers. Functions provided by block layer can be utilized by the file systems and the virtual memory manager to submit I/O requests to the block devices. These requests are transformed by the I/O scheduler to the low-level device driver. Red Hat Enterprise Linux AS 4 and SUSE Linux Enterprise Server 10 support four different I/O schedulers.

Deadline I/O scheduler

The Deadline I/O scheduler incorporates a per-request expiration-based approach and operates on five I/O queues. The basic idea behind I/O scheduler is that all read requests are satisfied within a specified time period. On the other hand, write requests do not have any specific deadlines. Web servers are found to perform better when configured with Deadline I/O scheduler and ext3.

Tunable parameters are listed in Table 7-6.

Table 7-6 Deadline I/O Scheduler Tunable parameters

Tunable parameter	Analysis
read_expire(in milliseconds)	Each read request that enters the scheduler is assigned a deadline factor that consists of current time plus the read_expire (less the read_expire earlier are the read requests serviced).
fifo_batch	Governs the number of requests moved to the dispatch queue for batch processing. Normally, I/O requests can be from continuous blocks or scattered blocks. The more the number of fifo_batch, the less the cost is for each I/O request.
seek_cost	The cost of a seek operation is per stream_unit basis, which is in kilobytes. The stream_unit dictates the number of kilobytes used to describe a single steam unit, for example, 4 KB. If a request consists of 20 KB, the actual cost is (20 + 4(steam_unit) -1)/4(steam_unit) = 6.
write_starved	Expressed in number of dispatches. It indicates the number of time I/O scheduler assigns preference to read over write requests.
front_merges	Controls the request merge technique. Based on the way files are laid out, back merge operations are more common than front merges. Setting the front_merges flag to 0 disables functionality.

You can see the Deadline I/O scheduler default parameter in the Example 7-4.

Example 7-4 Default tunable parameter for Deadline I/O scheduler

```
#cat /sys/block/sda/queue/iosched/read_expire
500
#cat /sys/block/sda/queue/iosched/write_expire
5000
#cat /sys/block/sda/queue/iosched/fifo_batch
16
#cat /sys/block/sda/queue/iosched/front_merges
1
#cat /sys/block/sda/queue/iosched/writes_starved
2
```

You can change the Deadline I/O scheduler default parameter read_expire to a lower value of 250 to improve performance of read requests, as shown in Example 7-5.

Example 7-5 Shows tuning of the default parameter for Deadline I/O scheduler

```
#echo 250 > /sys/block/sda/queue/iosched/read_expire
#cat /sys/block/sda/queue/iosched/read_expire
250
```

Noop I/O scheduler

The Noop I/O scheduler is an I/O scheduler that performs and provides basic merging and sorting functions. In large I/O subsystems that incorporate RAID controllers and a vast number of contemporary physical disk drives called TCQ drives, the Noop I/O scheduler has the potential to outperform the other three I/O schedulers as the workload increases.

Completely Fair Queuing I/O Scheduler

The Completely Fair Queuing (CFQ) I/O scheduler is implemented on the concept of fair allocation of I/O bandwidth among all the initiators of I/O requests. The CFQ I/O scheduler strive to manage per-process I/O bandwidth and provide fairness at the level of process granularity. The sequential writes perform better when CFQ I/O scheduler is configured with XFS. The number of requests fetched from each queue is controlled by the cfq_quantum tunable parameter.

You can see the CFQ I/O scheduler default parameters as shown in Example 7-6.

Example 7-6 Shows default tunable parameter for CFQ I/O scheduler

```
#cat /sys/block/sda/queue/iosched/queued
8
#cat /sys/block/sda/queue/iosched/quantum
4
```

You can change the CFQ I/O scheduler default parameter as shown in the Example 7-7.

Example 7-7 Shows tuning of the default tunable parameter for CFQ I/O scheduler

```
#echo 4 > /sys/block/sda/queue/iosched/queued
#cat /sys/block/sda/queue/iosched/queued
4
```

Anticipatory I/O scheduler

The Anticipatory I/O scheduler attempts to reduce the per-thread read response time. It introduces a controlled delay component into the dispatching equation. The delay is invoked on any new read request to the device driver. File servers are found to perform better when the Anticipatory I/O scheduler is configured with ext3. Sequential reads perform better when the Anticipatory I/O scheduler is configured with XFS or ext3. The tunables for the Anticipatory I/O scheduler are shown in Table 7-7.

Table 7-7 Anticipatory I/O scheduler tunable parameters

Tunable parameter	Analysis
read_expire	Governs the time frame until a read request is labeled as expired.
read_batch_expire(usually multiple of read_expire)	Governs the time assigned to a batch or set of read requests prior to serving any pending write request. A higher value increases the read request priority.
write_expire	Governs the time frame until a write request is labeled as expired.
write_batch_expire(usually multiple of write_expire	Governs the time assigned to a batch or set of write requests prior to serving any pending read requests. A higher value increases the write request priority.

Tunable parameter	Analysis
antic_expire	Controls the maximum amount of time the AS scheduler will idle before moving on to another request. The literature suggests initializing the parameter slightly higher for larger seek time devices.

You can see the Anticipatory I/O scheduler default parameters in Example 7-8.

Example 7-8 Shows default tunable parameter for Anticipatory I/O scheduler

```
#cat /sys/block/sda/queue/iosched/read_expire
120
#cat /sys/block/sda/queue/iosched/read_batch_expire
500
#cat /sys/block/sda/queue/iosched/write_expire
250
#cat /sys/block/sda/queue/iosched/write_batch_expire
120
#cat /sys/block/sda/queue/iosched/antic_expire.
10
```

You can change the Anticipatory I/O scheduler default parameter write_batch_expire to a higher value for better read performance, as shown in Example 7-9.

Example 7-9 Tuning the default parameter for the Anticipatory I/O scheduler

```
#echo 1000 > /sys/block/sda/queue/iosched/write_batch_expire
#cat /sys/block/sda/queue/iosched/write_batch_expire
1000
```

Setting I/O schedulers for block devices

It has been found that certain I/O schedulers perform better if configured for different file systems. Now it is possible to set different I/O schedulers for different block devices.

For example, you can set the CFQ scheduler for the system as the default, but use a specific device to use the Anticipatory or Noop schedulers, which can improve a device's throughput.

You can see which I/O scheduler is set for specific device, as shown in Example 7-10.

Example 7-10 Current configured I/O scheduler for a specific block device

```
#cat /sys/block/hda/queue/scheduler
noop anticipatory deadline [cfq]
```

- For SUSE Linux Enterprise Server 10

  You can set a specific I/O scheduler, as shown in the Example 7-11.

Example 7-11 Setting the Anticipatory I/O scheduler for hda

```
#echo anticipatory > /sys/block/hda/queue/scheduler
#cat /sys/block/hda/queue/scheduler
noop [anticipatory] deadline cfq
```

- For Red Hat Enterprise Linux AS 4

  You cannot change the scheduler during the runtime. You must specify a specific I/O scheduler during boot time or specify the I/O scheduler in the /etc/yaboot.conf and reboot the system. The default Red Hat Enterprise Linux AS 4 update 4 /etc/yaboot.conf is shown in Figure 7-8.

Figure 7-8 Shows Red Hat Enterprise Linux AS-4 default /etc/yaboot.conf

To change the default I/O scheduler, you need to edit /etc/yaboot.conf and add elevator=anticipatory to the append line, as shown in Figure 7-9, and then reboot the system.

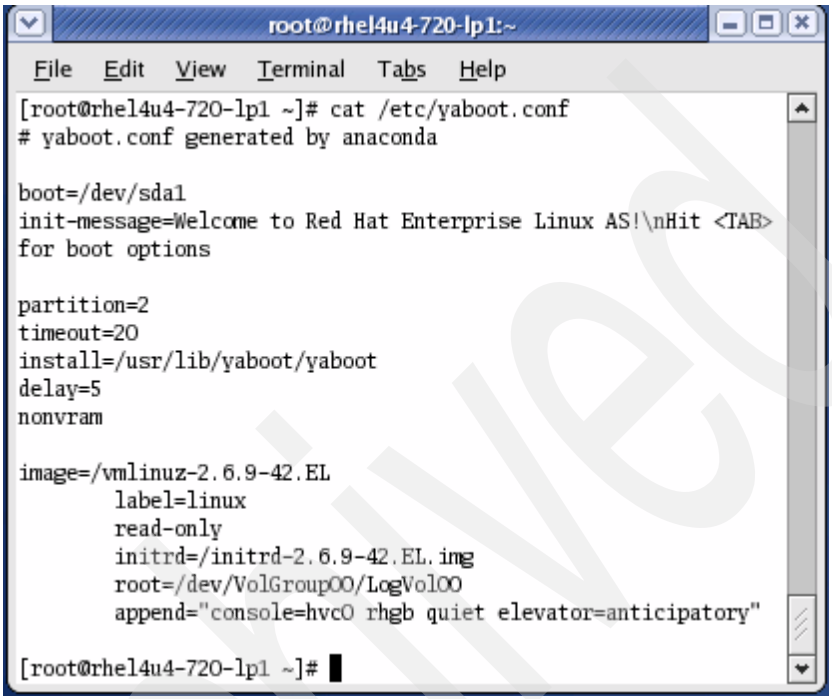


Figure 7-9 Shows /etc/yaboot.conf with anticipatory I/O scheduler.

7.7.2 Tuning file systems

Here we discuss tuning the file systems.

ext3

The ext3 file system provides high availability without impacting the robustness, simplicity, and reliability of ext2. ext3 is a minimal extension to ext2 to add support for journaling. If you want to adopt a journaling file system but do not have free partitions on your system, ext3 could be the journaling file system to use. ext3 can be used in Red Hat Enterprise Linux Server AS 4 and SUSE Linux Enterprise Server 10.

There are four ways to tune an ext3 file system:

- When the file system is created, which is the most efficient way. An ext3 file can be created using mke2fs or mkfs.ext3 or mkfs -t ext3, as shown in the Example 7-12. Some of the key options that are available with the **mkfs** command:
  - The -b block-size option specifies the block size in bytes (1024, 2048, or 4096 bytes per block).
  - The -N number-of-inodes option specifies the number of inodes.
  - The -T fs-type option specifies how the file system will be used. Valid options are:
    - new creates one inode per 4 KB block
    - largefile creates one inode per megabyte
    - largefile4 creates one inode per 4 MB

Example 7-12 Usage of mkfs to create ext3 file system

```
#mkfs.ext3 /dev/sdb1

mke2fs 1.28 (14-Jul-2006)
Filesystem label=
OS type: Linux
Block size=1024 (log 0)
Fragment size=1024 (log 0)
128016 inodes, 511984 blocks
```

- Through the tuning utility **tune2fs**, which can be used to tune the file system after it has been created. For example, you can eliminate the automatic checking using the **tune2fs** command -c option by setting checking to zero, as shown in the Example 7-13.

Example 7-13 Usage of tune2fs to eliminate automatic checking

```
#tune2fs -c 0 /dev/sdb1
tune2fs 1.28(14-Jul-2006)
Setting maximal mount count to -1
```

- Using a separate journal device on an ext3 file sytem.

  The first thing you need to do to use an external journal for an ext3 file system is to issue the **mkfs** command on the journal device. The block size of the external journal must be the same block size as the ext3 file system. For example, the /dev/hda1 device is used as the external log for ext3 file system, as shown below:

  ```
  #mkfs.ext3 -b 4096 -O journal_dev /dev/hda1
  #mkfs.ext3 -b 4096 -J device=/dev/hda1 /dev/hdb1
  ```

- Changing the default data mode during the mounting of the file system. There are three different data modes: writeback, ordered, and journal.

  – writeback mode

    In writeback mode, ext3 does not journal data at all. This mode provides a similar level of journaling as that of XFS, JFS, and ReiserFS in its default mode - metadata journaling. A crash+recovery can cause incorrect data to appear in files that were written shortly before the crash. This mode will typically provide the best ext3 performance.

  – ordered mode

    In ordered mode, ext3 only officially journals metadata, but it logically groups metadata and data blocks into a single unit called a transaction. When it is time to write the new metadata out to disk, the associated data blocks are written first. In general, this mode performs slightly slower than writeback, but significantly faster than journal mode.

  – journal mode

    In journal mode, ext3 provides full data and metadata journaling. All new data is written to the journal first, and then to its final location. In the event of a crash, the journal can be replayed, bringing both data and metadata into a consistent state. This mode is the slowest except when data needs to be read from and written to disk at the same time where it outperforms all other modes.

  You can change the default data mode during mounting of the file system as show below:

  ```
  #mount /dev/hda1 /data -o data=writeback
  ```

Next-Generation File System

Next-Generation File System (XFS) is 64-bit file system, designed to scale and have high performance. XFS uses many of the same techniques available in JFS. XFS supports metadata journaling and extremely large disk farms. A single XFS file system is designed to be 18,000 PB and a single file can be 9,000 PB. Currently, Linux supports 32-bit inode numbers, so this limits XFS to a 320-bit inode number. A single file system can in theory be as large as 18 million TB.

Currently, Linux has a 16 TB limit for 2.6 kernel. XFS is enabled only on SUSE Linux Enterprise Server 10.

The performance of XFS can be tuned using the following options:

- Increase the *inode size* while file system is created. The default size is 256 bytes, but can be increased up to 4 KB. This helps directories to keep more contents in the inode and need less disk I/O; however, inodes conversely need more I/O to read. The inode size can be increased as shown below:

  ```
  #mkfs -t xfs -i size=512 -f /dev/hdb1
  ```

- Decrease the *allocation group count* while the file system is created. For large file systems, keep the agcount as low as possible. An allocation group can be up to 4 GB in size; more allocation groups means more of them to scan in low free-space conditions. agcount can be set to 18, as shown below:

  ```
  #mkfs -t xfs -d agcount=18 -f /dev/hdb1
  ```

- Using *external log* for XFS: An external log improves performance because the log updates are saved to a different partition than their corresponding file system. In the following example, /dev/hda1 and /dev//hdb1 are spare partitions. The /dev/hda1 partition is used as the external log, as shown in the command below:

  ```
  #mkfs -t xfs -l logdev=/dev/hda1 /dev/hdb1
  ```

- Forcing the metadata to *sync* as an option while mounting the file system: The osyncisdsync option indicates that only the metadata are written back to disk similar to O_DSYNC, as shown in Example 7-14.
- Setting the *number of log buffers* that are held in memory as an option during mounting the file system, as shown in Example 7-14.
- Setting the *log buffers size* held in memory as an option during mounting the file system, as shown in Example 7-14.
- Disabling *access time* updates as an option during mounting the file system, as shown in Example 7-14.

Example 7-14 XFS mount example

```
#mount -t xfs -o
osyncisdsync,logbufs=8,logbsize=32768b,noatime/dev/hda1 /mnt/home1
```

- Increase the *log size* while file system is created. For metadata-intensive workload, the default log size could be the limiting factor that reduces the file system's performance. Better results are achieved by creating file systems with a larger log size, as shown below:

  ```
  #mkfs -t xfs -l size=32768b -f /dev/hdb1
  ```

Tip: It is a good idea to mount metadata-intensive file systems with the following options:

```
mount -t xfs -o logbufsize=8, logbzise=32768b /dev/device /mntpoint
```

Reiserfs

Reiser4 fs is a file system based on dancing tree algorithms. Originally designed by Hans Reiser, Reiserfs carries the analogy between databases and file systems to its logical conclusion. In essence, Reiserfs treats the entire disk partition as though it were a single database table. Directories, files, and file metadata are organized in an efficient data structure called a *balanced tree*. This differs somewhat from the way in which traditional file systems operate, but it offers large speed improvements for many applications, especially those that use lots of small files.

Reiserfs uses its balanced trees to streamline the process of finding the files and retrieving their security (and other) metadata. For extremely small files, the entire file's data can actually be stored physically near the file's metadata, so that both can be retrieved together with little or no movement of the disk seek mechanism. If an application needs to open many small files rapidly, this approach significantly improves performance. Another feature of Reiserfs is that the balanced tree stores not just metadata, but also the file data itself. Reiserfs is the default file system for SUSE Linux Enterprise Server 10.

Note: Reiserfs is not supported by Red Hat Enterprise Linux AS 4.

The performance of Reiserfs can be tuned using the following options:

- Choosing an appropriate hash function [r5, rupasov, tea, detect], while mounting the reiserfs file system. This specifies the hashing algorithm used to locate and write files within directories.

  The *rupasov* hashing algorithm is a fast hashing method that places and preserves locality, lexicographically mapping close file names to the close hash values.

  The *tea* hashing algorithm is a Davis-Meyer function that creates keys by thoroughly permuting bits in the name. It gets high randomness and, therefore, low probability of hash collision, but this entails performance costs. This hashing is a good section for large directories causing EHASHCOLLISION with r5 hash.

  THe *r5* hashing algorithm is a modified version of the rupasov hash with a reduced probability of hashing collisions. This is the default hashing algorithm.

The *detect* option instructs mount to detect the hash function in use by the instance of the file sytem being mounted. It writes this information into the super block. This option is useful only on the first mount of an old file system.

You can change the default hash function as shown below:

```
#mount -t reiserfs -o hash=tea /dev/sdb2 /mnt/reiserfs
```

- Disabling the *nolog* option during mounting the file system. It also provides a slight performance improvement in some situations at the cost of forcing **fsck** if the file system is not cleanly shut down. You can disable the nolog option as shown below:

  ```
  mount -t reiserfs -o nlog /dev/sdb2 /mnt/reiserfs
  ```

- Disable the *notail* option during mounting of the file system. It disables the packing of files into the tree. By default, Reiserfs stores small files and "file tails" directly into the tree. You can disable the notail option as shown below:

  ```
  mount -t reiserfs -o notail /dev/sdb2 /mnt/reiserfs
  ```

Tip: You can specify both mount options, for example, the notail and nolog options, while mounting the file system:

```
mount -t reiserfs -o nolog,notail /dev/sdc1 /fs1
```

- Specifying an *external log* device while creating the file system. An external log improves performance because the log updates are saved to a different partition than the log for the corresponding file system. This reduces the number of disk seeks.You can specify an external log device /dev/hda1 as shown below:

  ```
  mkreiserfs -j /dev/hda1 /dev/hdb1
  ```

7.7.3 Tuning file synchronization

On Linux systems, read and write requests are cached in memory. This cache is known as buffer/page cache. They avoid the actual transfer of data to the disk until the buffer is full or until applications call a sync function to flush the buffer/page cache. This strategy increases performance by avoiding the relatively slow mechanism process of writing to disk more often than necessary. Read and write requests can be classified as:

- Asynchronous I/O, which frees the application to perform other tasks while input is written or read. Asynchronous I/O permits efficient overlap of CPU and I/O processing, which can dramatically increase the performance of demanding applications.

Note: Only applications can perform asynchronous I/O using any open file descriptor.

- Synchronous I/O, which performs the write or read operation and verifies its completion before returning.

  Synchronized I/O is useful when the integrity of data and files is critical to an application. Synchronized output ensures that the data that is written to a device is actually stored there. Synchronized input ensures that the data that is read from a device is the latest updated data on that device.

Note: Synchronized I/O can degrade system performance.

Calling the pdflush at regular intervals is an acceptable way of synchronizing the data.

Tuning swap partition

The swap device is used when physical RAM is fully in use and the system needs additional memory. When there is no free memory available on the system, it begins paging the least-used data from memory to the swap areas on the disks. The initial swap partition is created during the Linux installation process with current guidelines stating that the size of the swap partition should be twice that of the physical RAM. Linux kernels 2.4 and beyond support swap sizes up to 24 GB per partition with an 8 TB theoretical maximum for 32-bit systems. Swap partitions should reside on separate disks.

If more memory is added to the server after the initial installation, additional swap space must be configured. There are two ways to configure additional swap space after the initial install:

- A free partition on the disk can be created as a swap partition. This can be difficult if the disk subsystem has no free space available. In that case, a swap file can be created.
- If there is a choice, the preferred option is to create additional swap partitions. There is a performance benefit because I/O to the swap partitions bypasses the file system and all of the work involved in writing to a file.

Another way to improve the performance of swap partitions or files is to create multiple swap areas. Linux can take advantage of multiple swap partitions or files and perform the reads and writes in parallel to the disks. After creating the additional swap partitions or files, the /etc/fstab file will contain such entries as shown in Example 7-15.

Example 7-15 /etc/fstab file

```
/dev/sda2       swap                    swap    sw              0 0
/dev/sdb2       swap                    swap    sw              0 0
/dev/sdc2       swap                    swap    sw              0 0
/dev/sdd2       swap                    swap    sw              0 0
```

Under normal circumstances, Linux would use the /dev/sda2 swap partition first, then /dev/sdb2, and so on, until it had allocated enough swapping space. This means that perhaps only the first partition, /dev/sda2, would be used if there is no need for a large swap space.

Spreading the data over all available swap partitions improves performance because all read/write requests are performed simultaneously to all selected partitions. If you change the file as shown in Example 7-16, you will assign a higher priority level to the first three partitions.

Example 7-16 Modified /etc/fstab to shown parallel swap partitions

```
/dev/sda2       swap                    swap    sw,pri=3        0 0
/dev/sdb2       swap                    swap    sw,pri=3        0 0
/dev/sdc2       swap                    swap    sw,pri=3        0 0
/dev/sdd2       swap                    swap    sw,pri=1        0 0
```

Swap partitions are used from the highest priority to the lowest (where 32767 is the highest and 0 is the lowest). Giving the same priority to the first three disks causes the data to be written to all three disks; the system does not wait until the first swap partition is full before it starts to write on the next partition. The system uses the first three partitions in parallel and performance generally improves.

The fourth partition is used if additional space is needed for swapping after the first three are completely filled up. It is also possible to give all partitions the same priority to stripe the data over all partitions, but if one drive is slower than the others, performance will decrease. A general rule is that the swap partitions should be on the fastest drives available.

7.8 Tuning the network subsystem

Most of applications utilize the networking infrastructure provided by the Linux, which makes networking performance an important consideration for system administrators. Most of the networking parameters are set to default values during the system boot time. The defaults are usually a good choice for most of the applications most of the time. Also, some of the networking parameters are tuned automatically by the kernel itself during the , such as the size of the socket buffer in use. But there are situations where the default values and dynamic tuning is insufficient, and adequate or improved performance requires the manual tuning of kernel parameters.

7.8.1 Disabling unwanted networking services

Some of the networking parameters are enabled by default during the system bootup. These parameters can be disabled if required by the system administrator to boost the network performance to some extent. The following **sysctl** commands are used primarily to change the default values.

- To ignore redirects from machines that are listed as gateways, use **sysctl** as shown below. Redirect can be used to perform attacks, so we only want to allow them from trusted sources.

```
#sysctl -w net.ipv4.conf.eth0.secure_redirects=1
#sysctl -w net.ipv4.conf.lo.secure_redirects=1
#sysctl -w net.ipv4.conf.default.secure_redirects=1
#sysctl -w net.ipv4.conf.all.secure_redirects=1
```

- The ICMP redirect is a mechanism for routers to convey routing information to hosts. For example, the gateway can send a redirect message to a host when the gateway receives an internet datagram from a host on a network to which the gateway is attached. The gateway checks the routing table to get the address of the next gateway, and the second gateway will route the internet datagram to a destination in the network. Disable these redirects using the following commands:

```
#sysctl -w net.ipv4.conf.eth0.accept_redirects=0
#sysctl -w net.ipv4.conf.lo.accept_redirects=0
#sysctl -w net.ipv4.conf.default.accept_redirects=0
#sysctl -w net.ipv4.conf.all.accept_redirects=0
```

- If this server does not act as a router, then it does not need to send redirects, so they can be disabled.

```
#sysctl -w net.ipv4.conf.eth0.send_redirects=0
#sysctl -w net.ipv4.conf.lo.send_redirects=0
#sysctl -w net.ipv4.conf.default.send_redirects=0
#sysctl -w net.ipv4.conf.all.send_redirects=0
```

- Configure the server to ignore broadcast pings or smurf attacks.

```
#sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

- Ignore all kinds of ICMP packets or pings.

```
#sysctl -w net.ipv4.icmp_echo_ignore_all=1
```

- Some routers send invalid responses to broadcast frames, and each one generates a warning that is logged by the kernel. These responses can be ignored.

```
#sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
```

7.8.2 Tuning TCP buffers

Here we discuss tuning the TCP buffers.

tcp_rmem

This parameter is a vector of 3 INTEGERs: min, default, and max. It controls the initial size of the read buffer in bytes when a socket is created. The larger the receiver socket buffer, the larger the window advertised by TCP and the faster the sender can send data. The default tcp_rmem value is shown in the Example 7-17.

Example 7-17 Shows the default tcp_rmem values.

```
#sysctl net.ipv4.tcp_rmem
#net.ipv4.tcp_rmem = 4096 87380 174760
```

- min

  Minimal size of receive buffer used by TCP sockets. It is guaranteed for each TCP socket, even under moderate memory pressure. The default is 8 KB.

- default

  The default size of the receive buffer used by TCP sockets. This value overrides net.core.rmem_default used by other protocols. The default is 87380 bytes. This value results in a window of 65535 with a default setting of tcp_adv_win_scale and tcp_app_win:0 and a bit less for default tcp_app_win.

- max

  The maximum size of the receive buffer allowed for automatically selected receiver buffers for the TCP socket. This value does not override net.core.rmem_max, “static” selection, so SO_RCVBUF does not use this. The default is 87380*2 bytes.

You can change the default value as shown below:

```
#sysctl -w net.ipv4.tcp_rmem=”32768 436600 873200"
```

tcp_wmem

This parameter is a vector of 3 INTEGERs: min, default, and max. It controls the memory allocated for send buffers. Having a large buffer is beneficial in that it allows the applications to transfer a large amount of data to the write buffer without blocking. The default tcp_wmem value is shown in Example 7-18.

Example 7-18 Default tcp_wmem values

```
#sysctl net.ipv4.tcp_wmem
#net.ipv4.tcp_wmem = 4096 16384 131072
```

- min

  Amount of memory reserved for the send buffers for the TCP socket. Each TCP socket has rights to use it based on its origin. The default is 4 KB.

- default

  Amount of memory allowed for the send buffers for the TCP socket by default. This value overrides net.core.wmem_default used by other protocols, and it is usually lower than net.core.wmem_default. The default is 16 KB.

- max

  The maximum amount of memory allowed for the automatically selected send buffers for the TCP socket. This value does not override net.core.wmem_max, so “static” selection through SO_SNDBUF does not use this. The default is 128 KB.

You can change the default value as shown below:

```
#sysctl -w net.ipv4.tcp_wmem=”8192 436600 873200"
```

tcp_mem

This parameter is also a vector of 3 INTEGERs: min, pressure, and max. It governs the amount of memory pages allocated globally to sockets in the system. The default tcp_mem value is shown in the Example 7-19.

Example 7-19 Default tcp_mem values

```
#sysctl net.ipv4.tcp_mem
#net.ipv4.tcp_mem = 786432 1048576 1572864
```

- low

  Below this number of pages, TCP is not concerned about its memory needs.

- pressure

  When the amount of memory allocated by TCP exceeds this number of pages, TCP moderates its memory consumption and enters memory pressure mode, which is exited when memory consumption falls under “low”.

- high

  Number of pages allowed for queuing by all TCP sockets.

The defaults are calculated at boot time from the amount of available memory. You can change the default value as shown below:

```
#sysctl -w net.ipv4.tcp_mem="1000000 1048576 1572864"
```

7.8.3 Disabling TCP options

Here we discuss disabling the TCP options.

tcp_window_scaling

This parameter enables the window scaling feature to allow window sizes greater than 64 KB. For connections over satellite links and on networks with large round trip times, employing large windows (greater than 64 KB) results in a significant improvement in performance. Optimum throughput is achieved by maintaining a window at least as large as the bandwidth-delay product of the network. It should be noted that socket buffers larger than 64 KB are still potentially beneficial even when window scaling is turned off. You can disable them as shown below:

```
#sysctl -w net.ipv4.tcp_window_scaling=0
```

tcp_sack

This parameter enables the TCP selective acknowledgements feature. SACK can be beneficial when losses occur frequently and round-trip times are long. In environments like high-speed local networks with very short round-trip times and negligible loss, performance can actually be improved by turning SACK off. This will avoid the need to process SACK options.You can disable them as shown below:

```
#sysctl -w net.ipv4.tcp_sack=0
```

tcp_dsack

This parameter enables TCP D-SACK features, which is an enhancement to SACK to detect unnecessary retransmits. It should be disabled if SACK is disabled.You can disable the features as shown below:

```
#sysctl -w net.ipv4.tcp_dsack=0
```

tcp_fack

This parameter enables the TCP forwarding acknowledgement feature. It is a refinement of the SACK protocol to improve congestion control in TCP. It should also be disabled if SACK is disabled.You can disable it as shown below:

```
#sysctl -w net.ipv4.tcp_fack=0
```

7.8.4 Tuning TCP connection management

TCP is a connection oriented protocol. Tuning these parameters can have an impact on the number of connections that it can support simultaneously, which is an important consideration for busy servers.

tcp_max_syn_backlog

This parameter controls the maximal number of remembered connection requests, which still did not receive an acknowledgment from connecting client. The default value is 1024 for systems with more than 128 Mb of memory, and 128 for low memory machines. If clients experience failures connecting to busy servers, this value could be increased. You can increase the value as shown below:

```
#sysctl -w net.ipv4.tcp_max_syn_backlog=2048
```

tcp_synack_retries

This parameter controls the number of times SYNACKs for a passive TCP connection attempt will be retransmitted. It should not be higher than 255. The default value is five, which corresponds to about 180 seconds. Reducing this number results in earlier detection of a failed connection attempt from the remote host. You can decrease the value as shown below:

```
#sysctl -w net.ipv4.tcp_synack_retries=2
```

tcp_retries2

This parameter controls how may times to retry before killing an alive TCP connection. RFC1122 says that the limit should be longer than 100 seconds, but that is too small a number. The default value of 15 corresponds to 13-30 minutes, depending on RTO. Reducing this number results in the earlier detection of a failed connection to the remote host. You can decrease the value as shown below:

```
#sysctl -w net.ipv4.tcp_retries2=10
```

tcp_keepalive_time

This parameter controls how often TCP sends keepalive messages, if keepalive is enabled. If the connection is idle for the number of seconds specified by this parameter, the kernel initiates the tcp_keepalive messages. The default is two hours. You can change it as shown below:

```
#sysctl -w net.ipv4.tcp_keepalive_time=3600
```

tcp_keepalive_intvl

This parameter specifies the time interval, in seconds, between the keepalive probes send by the kernel to the remote host. The default is 75 seconds, that is, connections will be aborted after two minutes of retries. You can change it as shown below:

```
#sysctl -w net.ipv4.tcp_keepalive_intvl =50
```

tcp_keepalive_probes

This parameter specifies the maximum number of keepalive probes the kernel sends to the remote host to detect if it is still alive. The default value is 9.

The higher the tcp_keepalive_time, tcp_keepalive_intvl, and tcp_keepalive probes values, the larger the delays and round-trip times are on the internet. It's a best practice to lower the preceding parameters to detect remote hosts that have gone away earlier. This minimizes the resources tied up in extinct connections.

You can change this parameter as shown below:

```
#sysctl -w net.ipv4.tcp_keepalive_probes =5
```

7.8.5 Tuning IP fragmentation

Here we discuss tuning the IP fragmentation.

ipfrag_high_thresh

This parameter controls the maximum memory allocated to reassemble IP fragments.

ipfrag_low_thresh

These values are used to reassemble IP fragments. The fragment handler will drop packets until the ipfrag_low_thres is reached.

Fragmentation occurs when there is an error during the transmission of TCP packets. Valid packets are stored in memory (as defined with these parameters) while corrupted packets are retransmitted. We should set the ipfrag parameters particularly for NFS and Samba servers.

For example, to set the range of available memory between 256 MB and 384 MB:

```
#sysctl -w net.ipv4.ipfrag_low_thresh=262144
```

8

File and printer server tuning

This chapter provides information about tuning for a System p server running Linux and acting as a file and printer server. Additionally, after tuning is done, a description of the modifications done are included to allow a system administrator to improve their system's overall performance.

The chapter begins with a brief introduction for file and printer serving, then a discussion of the effects that changes to the hardware capabilities has on the system response time. Next comes a brief discussion of the environment used for the test.

The last section of the chapter contains the details in performance gains or losses observed in the test environment and the list of changes that caused such changes.

8.1 File server workload

The role of the file server is to store, retrieve, and update data that depends on client requests. Therefore, the critical areas that impact performance are the speed of the data transfer and the networking subsystems. The amount of memory available to resources, such as network buffers and disk I/O caching, also greatly influence performance. Processor speed or quantity typically has little impact on file server performance.

In larger environments, consideration should also be given to where the file servers are located within the networking environment. It would be advisable to locate them on a high-speed backbone as close to the core switches as possible.

The subsystems that will have the most impact on file server performance are:

- Network

  The network subsystem, particularly the network interface card or bandwidth of the LAN itself, may create a bottleneck due to heavy workload or the need to service large data transfers. Binding each network adapter to a CPU is a very good practice, although it depends on the number of network adapters, the number of CPUs, and the number of ports per network adapter. In 6.3.3, “Network performance indicators” on page 150, there is information about key parameters that should be monitored in order to identify a network bottleneck.

- Disk

  Since the purpose of a file server is to supply files to the client, the server must initially read all data from a disk. Thus, the disk subsystem is a potential bottleneck. Adjust the allocation unit size according to the size of the disk. If possible, keep the paging file, the operating system, and the data on separate physical disk drives. In 6.3.4, “Disk I/O performance indicators” on page 155, there is information about key parameters that should be monitored in order to identify a network bottleneck.

- Memory

  Insufficient memory may limit the ability to cache files and thus cause more disk activity, which slows down the system. You should have enough memory in your file server so that no paging to disk occurs. Avoid putting the paging file system on a RAID 5 array, and avoid placing multiple paging file systems on different partitions on the same physical drive(s). In 6.3.2, “Memory performance indicators” on page 148, there is information about key parameters that should be monitored in order to identify a memory bottleneck.

- Processor

  In the file server, the processor subsystem is least likely to cause a bottleneck. The CPU only becomes the bottleneck if the data is transferred from the network in small block sizes. In 6.3.1, “CPU performance indicators” on page 145, there is information about key parameters that should be monitored in order to identify a network bottleneck.

Tip: A common misconception is that CPU capacity is important. CPU is rarely a source of performance bottlenecks for file servers due to the nature of the workload.

When a client requests a file, the server must initially locate, then read and forward the requested data back to the client. The reverse of this applies when the client is updating a file. Therefore, the disk subsystem is potentially a bottleneck.

The network subsystem, particularly the network interface card or the bandwidth of the LAN itself, may create a bottleneck due to heavy workload or latency.

Insufficient memory may limit the ability to cache files and thus cause more disk activity, which will result in performance degradation.

There are several solutions to allow a system to act as a file server. However in a Linux context, two of the most relevant solutions at this time are:

- Samba
- Network File System

8.1.1 Samba

Samba is a popular open source free software program that allows users to access and use files, printers, and other commonly shared resources on a network. Samba is often referred to as a *Network File System* and can be installed on a variety of operating system platforms, including Linux.

Samba is based on the common client/server protocol of Server Message Block (SMB) and Common Internet File System (CIFS). Using client software that also supports SMB/CIFS (most commonly available in the Microsoft Windows operating system), a user sends a series of client requests to the Samba server on another computer in order to open that computer's files, access a shared printer, or access other resources. The Samba server on the other computer responds to each client request, either granting or denying access to its shared files and resources.

8.1.2 Network File System

The Network File System (NFS) is a client/server application that lets a computer user view and optionally store and update files on a remote computer as though they were on the user's own computer. The user's system needs to have an NFS client (most commonly available in Linux and most UNIX flavors operating system) and the other computer needs the NFS server. Most of the recent clients and servers require that the TCP/IP stack be installed and configured since the NFS server and client use TCP/IP as the protocols that send the files and updates back and forth. However, the User Datagram Protocol (UDP), which comes with TCP/IP, has been used instead of TCP with earlier versions of NFS client and server.

NFS was developed by Sun Microsystems and has been designated a file server standard. Its protocol uses the Remote Procedure Call (RPC) method of communication between computers. You can install NFS on Windows 95 and some other operating systems using products like Sun's Solstice™ Network Client.

Using NFS, the user or a system administrator can mount all or a portion of a file system (which is a portion of the hierarchical tree in any file directory and subdirectory, including the one you find on your PC or Mac). The portion of your file system that is mounted (designated as accessible) can be accessed with whatever privileges go with your access to each file (read-only or read-write).

8.2 The effect of server hardware on File Serving performance

Figure 8-1 shows the performance characteristics of a typical file server.

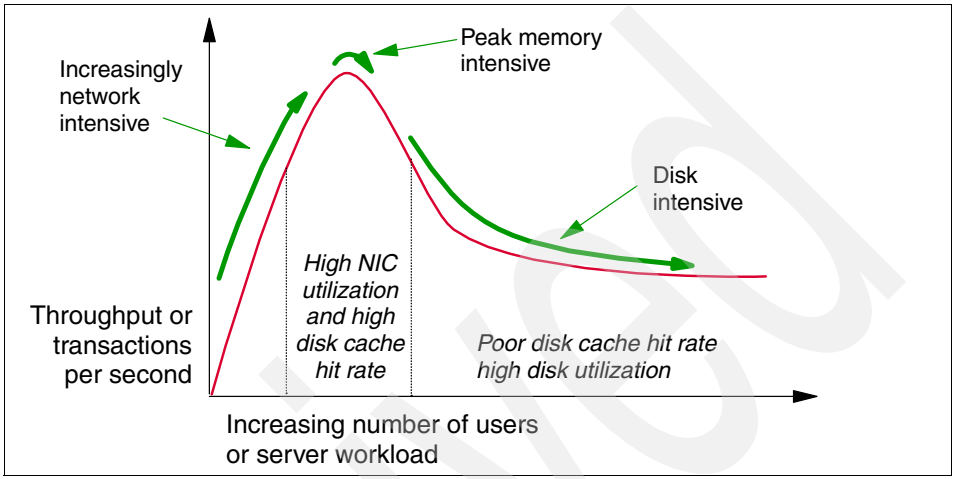


Figure 8-1 File server performance characteristics

As the number of users in the network or the server workload increases, the performance of the network can be considered network intensive. The throughput will increase until it reaches a limiting peak, and then it will taper off as the slower disk subsystem becomes apparent. The more transactions are processed, the more writing to disk will be required. As this becomes heavier, the performance will move into the disk-intensive phase, which will limit the server's performance.

Tests have shown that by using a faster network card, peak performance can be improved for lower loads, but throughput will be restricted at higher loads by the disk subsystem. This is shown in Figure 8-2.

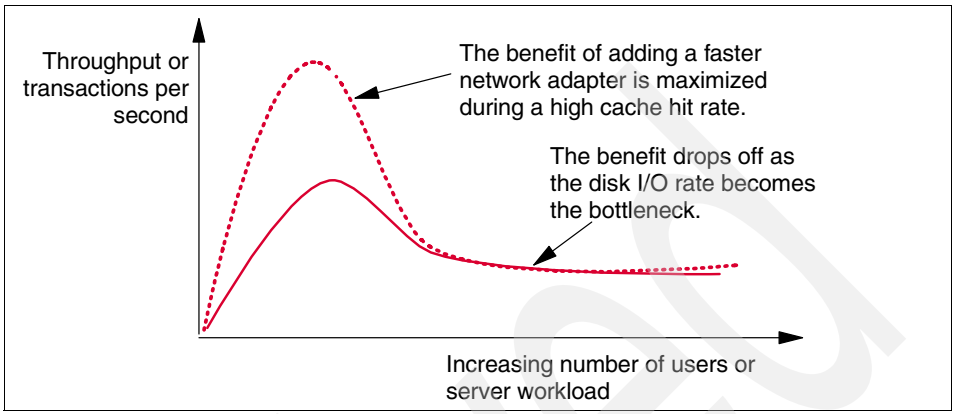


Figure 8-2 The effect of adding a faster network card

To determine if your network is a performance bottleneck, examine the performance counters listed in 6.3.3, “Network performance indicators” on page 150.

Using a faster disk subsystem (number of disks, mechanical speed, and disk cache), the peak throughput performance can be extended for higher server loads. This is shown in Figure 8-3. The upgrading of both network and disk subsystems will provide cumulative improvements.

To determine if your disk subsystem is a performance bottleneck, please review the tools and techniques listed in 6.3.4, “Disk I/O performance indicators” on page 155.

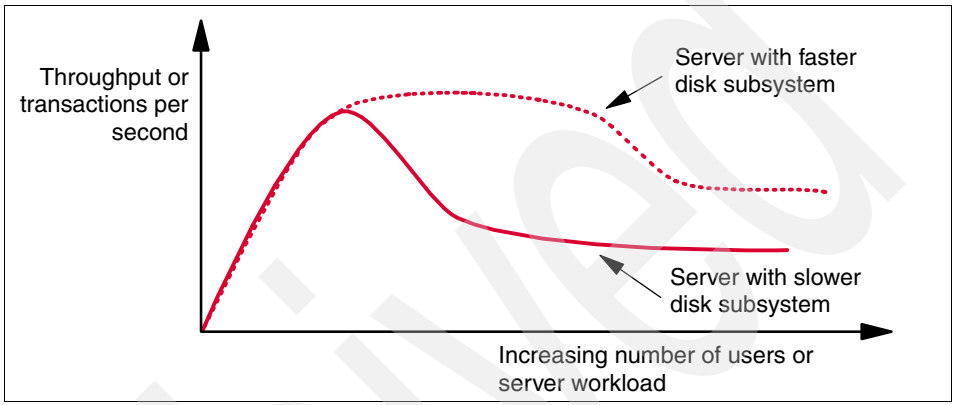


Figure 8-3 The effect of adding faster disk system

Increasing the RAM also helps by increasing the size of the file system cache. This also has the effect of extending the peak throughput for higher loads, as shown in Figure 8-4.

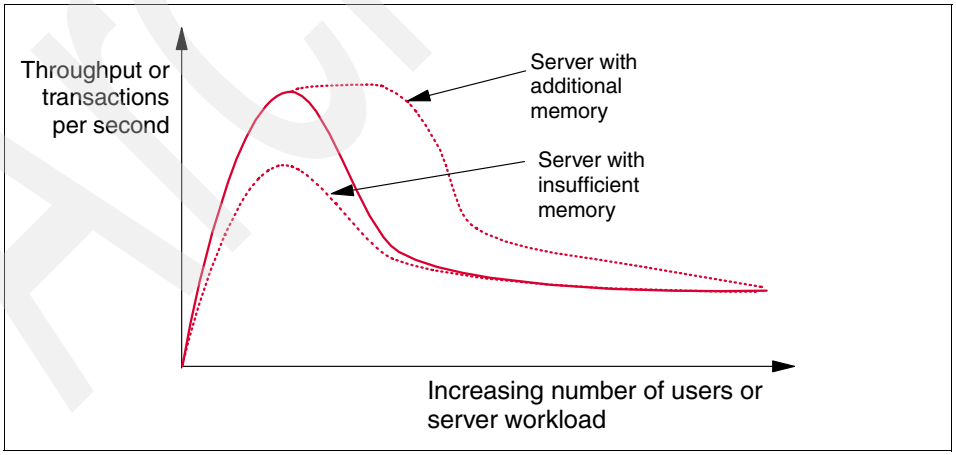


Figure 8-4 The effect of adding memory

To determine if your memory subsystem is a performance bottleneck, examine the Performance counters listed in 6.3.2, “Memory performance indicators” on page 148.

8.3 The file server testing environment

In order to test and tune a file server on a System p server running Linux, a stable and predefined testing environment must be present. With this type of environment, it is possible to compare performance data obtained from similar tests done with different tuning parameters at the operating system, application level, or both.

All the tests in this study were done using an OpenPower 720 server with four CPUs of 1.5 GHz, 32 GB of RAM, eight disks of 36 GB capable of creating micropartions as well as virtualize the I/O adapters and disks.

To simulate multiple clients, a BladeCenter chassis connected to the OpenPower 720 through a 100 Mbps Fast-Ethernet network was used. The BladeCenter chassis itself is populated with a mix of Intel and POWER blades that were configured to access the file servers within the OpenPower 720.

The Openpower 720 itself was configured with four partitions:

VIOS A partition controlling all physical I/O resources used to enable to share the physical resources by the other partitions. It is important to consider that in this partition one disk was used for VIOS install, and six disks were used to set up a RAID 5 array that was later carved into the target devices for Virtual SCSI clients to allow the other partitions to boot from the array. Finally, the last disk was left alone as a spare disk. This partition was configured with 0.7 CPU and left uncapped, as well as 768 MB of RAM. *During all tests, this partition was monitored to guarantee that it never suffered from any type of resource starvation.*

Client simulation A partition created to simulate clients from within the OpenPower 720 server. This partition allowed the tests to stress the disk system without any need to overcommit the VIOS server CPU resources or the 10/100 Mbps network. This was attained by using the Virtual LAN capabilities of the POWER5 systems to allow in memory network transfer at rates extremely higher than what is allowed through the existing LAN. Although this partition was configured with 0.1 CPU and 4 GB of RAM, the fact

that it was uncapped allowed it to grab the CPU cycles available at resources needed from the share CPU pool.

SUSE Server A partition created to install SUSE Linux Enterprise Server 10 with all the packages available in the install image and using the default parameters for all subsystems. The partition is used to measure simulated file serving workloads. The partition was configured with two CPUs, 4 GB of RAM, and 40 GB of disk mapped to a logical volume within the VIOS' RAID 5 array. It is important to keep in mind that this partition was set up as capped in order to make sure that it only had the cycles of two CPUs allocated to it.

Red Hat Server A partition created to install Red Hat Enterprise Linux AS 4 with all the packages available in the install image and using the default parameters for all subsystems. The partition is used to measure simulated file serving workloads. The partition was configured with two CPUs, 4 GB of RAM, and 40 GB of Disk mapped to a logical volume within the VIOS' RAID 5 array. It is important to keep in mind that this partition was set up as capped in order to make sure that it only had the cycles of two CPUs allocated to it.

A logical diagram of the environment described above can be seen in Figure 8-5.

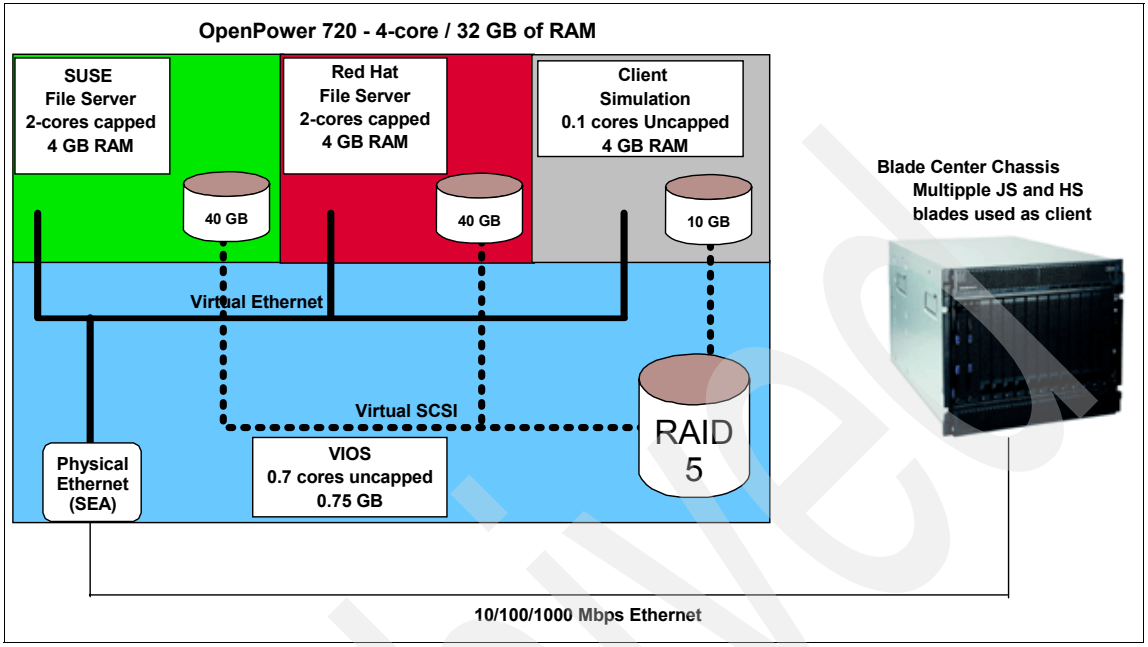


Figure 8-5 Logical diagram of the file system tuning environment

In each of the servers, a set of four files were created to be transferred from the server to the clients and vice-versa. These file vary in size to allow different aspects of file serving to be tested and compared.

Additionally, each client has received a set of scripts to simulate a continuous loop of either downloading or uploading one of the four files. Example 8-1 lists the simple script to simulate a read client for a file server used in the test.

Example 8-1 Script to simulate a client downloading a file from the server

```
#!/bin/bash

# Infinite loop downloading files from a server
# Pls., mount the files system into /home/metelo/sambascripts/mount
# at mount time you should pick the right server (Red Hat or SUSE)
# This script simulates a client downloading a 170MB file
# from a server
# pls input a file name for temporary file (To allow multiple clients)

echo 'Infinite loop downloading 170MB file from server'
echo 'Make sure to add a temporary filename as the parameter to allow'
echo 'multiple instances from the same server'
echo 'Ctrl + C to terminate'

Z=1
while (Z=1)

do if (test -f "/home/metelo/sambatestscripts/mount/170MB.data") then
  date
  cp /home/metelo/sambatestscripts/mount/170MB.data /home/metelo/$1
  rm -f /home/metelo/$1
else
  echo 'File not found'
  exit
fi
done
```

Example 8-2 shows the script to simulate a client writing the file to the server.

Example 8-2 Script to simulate a client uploading a file from the server

```
#!/bin/bash
# Infinite loop uploading files from a server
# pls, mount the files system into /home/metelo/sambascripts/mount
# at mount time you should pick the right server (Red Hat or SUSE)
# This script simulates a client downloading a 1MB file from a server
# pls input a file name for temporary file (To allow multiple clients)

echo 'Infinite loop uploading 40MB file from server'
echo 'Make sure to add a temporary filename as the parameter to allow'
echo 'multiple instances from the same server'
echo 'Ctrl + C to terminate'

Z=1
while (Z=1) do
  date
  cp /home/metelo/sambateststcripts/files/40MB.data
/home/metelo/sambateststcripts/mount/temp/$1
  rm -f home/metelo/sambatestscripts/mount/$1

done
```

The last and most important script listed here (Example 8-3) is responsible for allowing the measurement of how long a series of transfers (50% read and 50% write) and deletes took to complete. This script is intended to be used with a combination of the scripts described above to simulate a loaded server.

Example 8-3 Control script that allows to measure transfer time of files

```
#!/bin/bash
# Execute $1 copies to the current directory of file
# /home/metelo/sambatestcripts/$2
# Writes the start/end time
# pls, mount the files system into /home/metelo/sambascripts/mount
# at mount time you should pick the right server (Red Hat or SUSE)
# This script simulates a client downloading and uploading
# a XXMB file from a server
# Pls input a file name for temporary file (To allow multiple clients)
# A file named count is generated locally to allow monitor progress
# of long runs (for example 100 X downloading the 690MB file)

echo Copying [$2] file $1 times from the samba server
ls mount | grep Redhat
ls mount | grep SUSE

COUNTER=0
date
while [ $COUNTER -lt $1 ]; do
  cp /home/metelo/sambatestscripts/mount/$2 /home/metelo/$2
  rm -f /home/metelo/$2
  cp /home/metelo/sambatestscripts/files/$2
/home/metelo/sambatestscripts/mount/temp/$2
  rm -f /home/metelo/sambatestscripts/mount/temp/$2
  let COUNTER=COUNTER+1
  echo $COUNTER >> count
done
date
```

With the scripts listed above, it was possible to establish a starting metric of the transfer times for both the Red Hat Enterprise Linux and SUSE Linux Enterprise Server's default install.

8.4 Tuning Linux for file serving on a System p server

We started this task by looking at the default file systems and I/O schedulers.

8.4.1 Choosing a file system and an I/O scheduler

With the diverse options available for file systems and I/O schedulers available to run with Linux, as described in 7.7.1, “Tuning the disk I/O scheduler” on page 185 and in 7.7.2, “Tuning file systems” on page 192, we expect that the various combination of them would yield different performance results depending on the server hardware architecture, workload profile, and the I/O devices used by the solution.

This section does not look for a definite answer on which combination of a scheduler and a file system type will provide the best performance, but instead, it reports the results of tests done using an OpenPower 720 server running both SUSE Linux Enterprise Server and Red Hat Enterprise Linux in an environment, as described in 8.3, “The file server testing environment” on page 214.

When performing a default complete installation of Red Hat Enterprise Linux or SUSE Linux Enterprise Server editions, a system administrator will be presented with a different set of file systems available. This is a good indicator that different results should be expected when doing performance comparisons with the intent of picking a file system and an I/O scheduler for a System p server acting as a file server.

The most current SUSE Linux Enterprise Server install images already includes kernel support for the following file systems of interest for this study: ext3 and XFS, while the most current Red Hat Enterprise Linux install images only provides support for ext3.

The tests done were based on a series of read and writes (distributed at a 50/50 ratio) of files of different sizes (1 MB, 40 MB, 170 MB, and 690 MB). For both Red Hat Enterprise Linux and SUSE Linux Enterprise Server, the available file systems and schedulers were rotated in search of which combination would provide the best performance gain when compared to the distributions’ default choices.

An important part of the test was the System p servers Virtual LAN capabilities. It allowed the environment bottleneck to be moved from the fast-Ethernet network to the disk system available in the hard drives, as pre-existing measurements indicated that the Virtual LAN connectivity bandwidth outperformed the gigabit-Ethernet network.

The following charts demonstrate the relative performance changes measured as the file system and I/O schedulers were rotated for each of the file sizes listed above. The percent Gain/Loss is relative to the default OS installation. If a change was made and the number was 5% better, then the chart will show the number as being 105%.

SUSE Linux Enterprise Server measured performance gain/loss

The percent Gain/Loss is relative to the default OS installation. If a change was made and the number was 5% better, then the chart will show the number as being 105%. Figure 8-6 shows the outcome of the 1 MB tests.

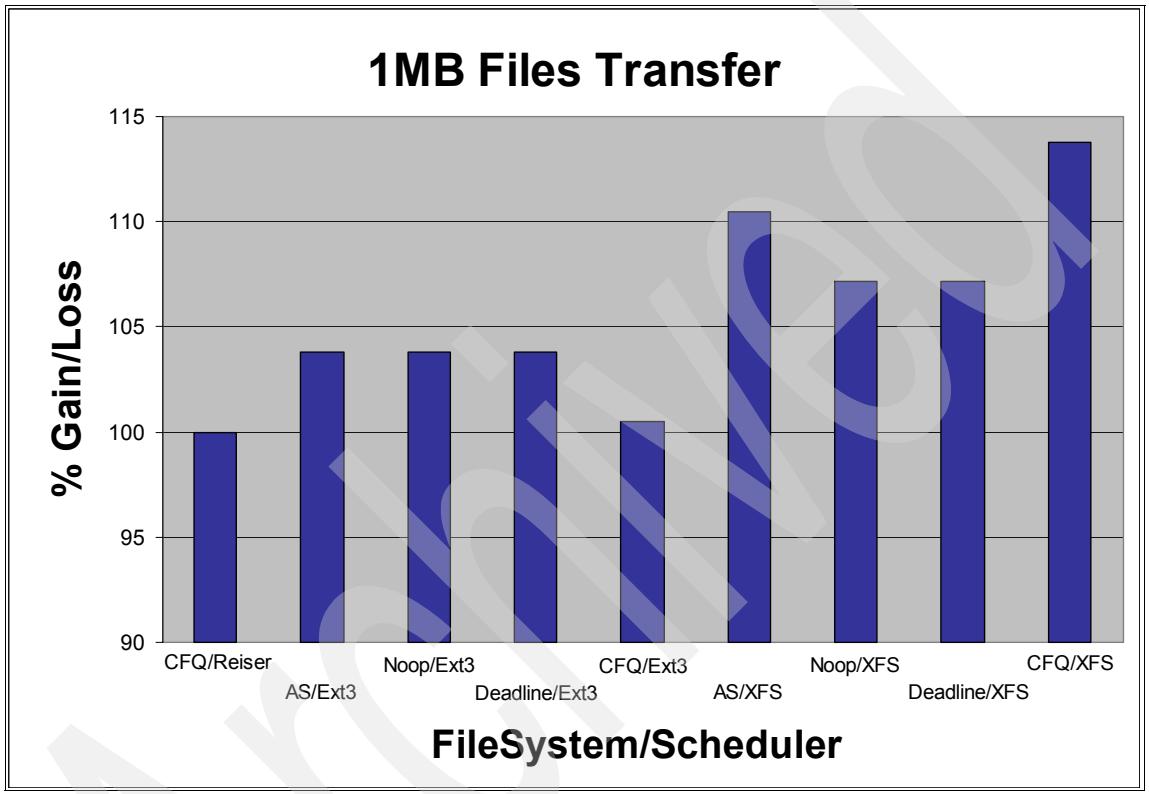


Figure 8-6 SUSE Linux Enterprise Server 1 MB file transfer times gains/loss

Figure 8-7 shows the outcome of the 40 MB tests with SUSE Linux Enterprise Server 10.

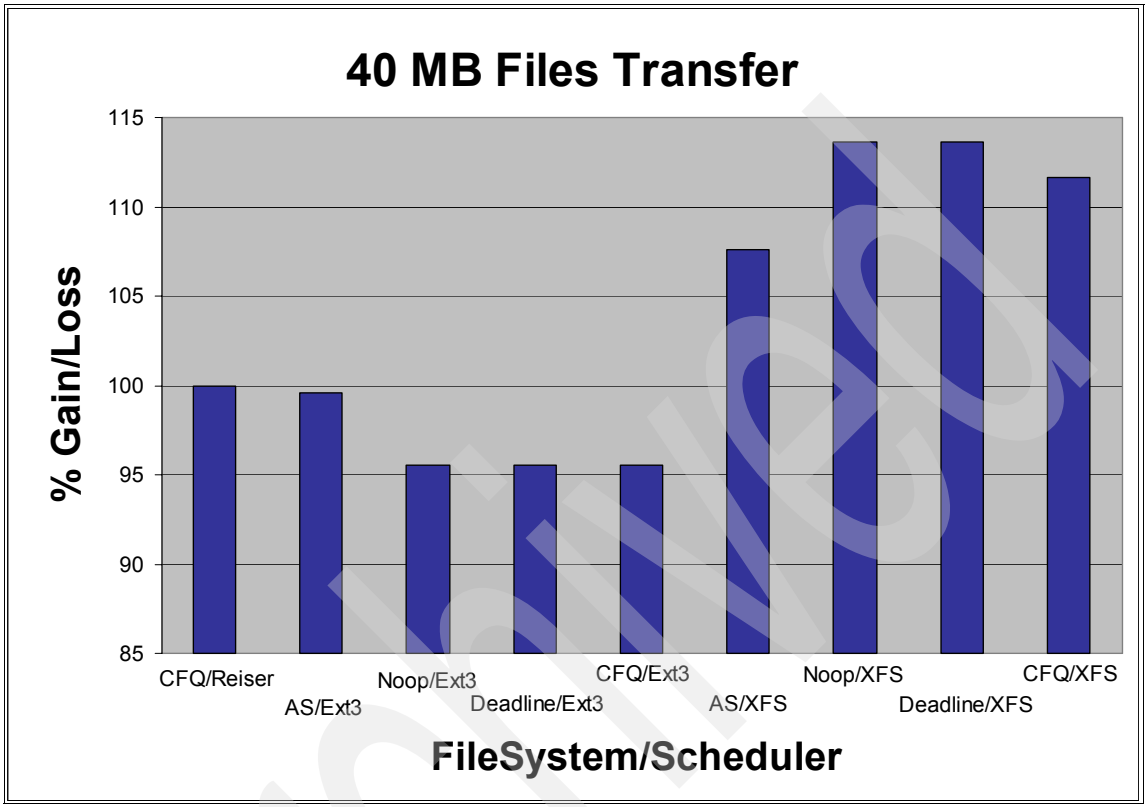


Figure 8-7 SUSE Linux Enterprise Server 40 MB file transfer times gains/loss

Figure 8-8 shows the outcome of the 170 MB tests with SUSE Linux Enterprise Server 10.

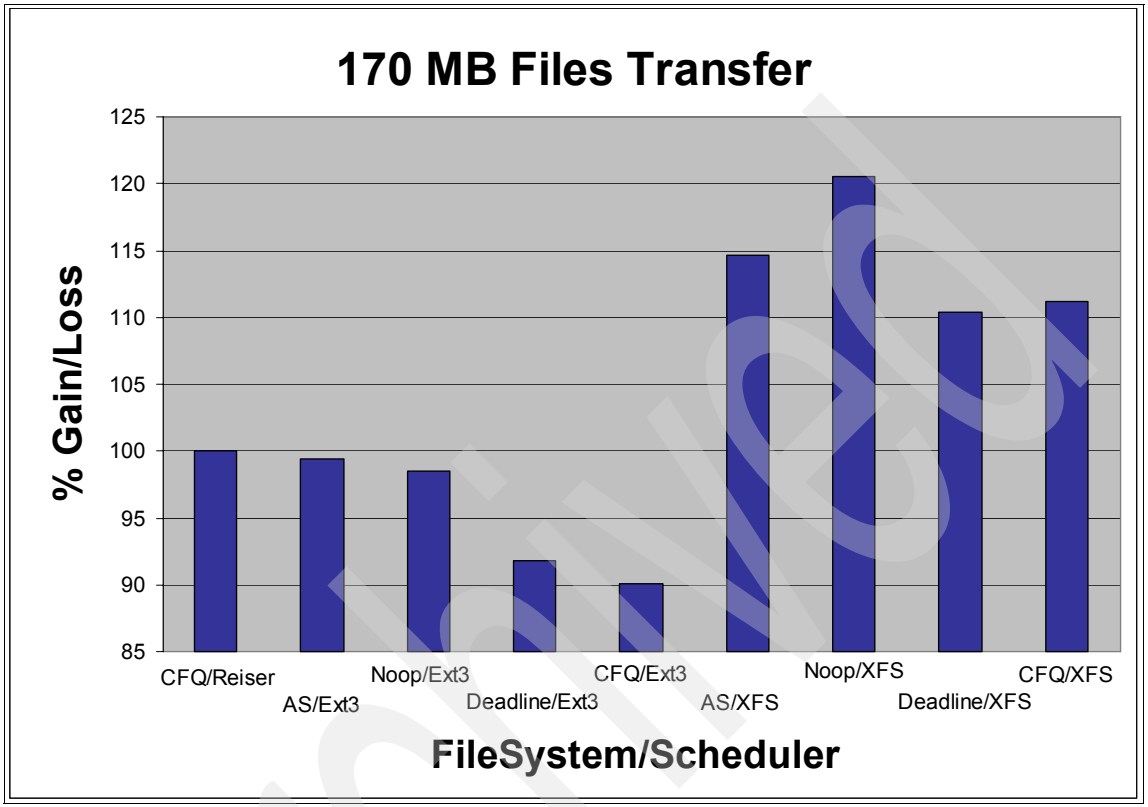


Figure 8-8 SUSE Linux Enterprise Server 170 MB file transfer times gains/loss

Figure 8-9 shows the outcome of the 690 MB tests with SUSE Linux Enterprise Server 10.

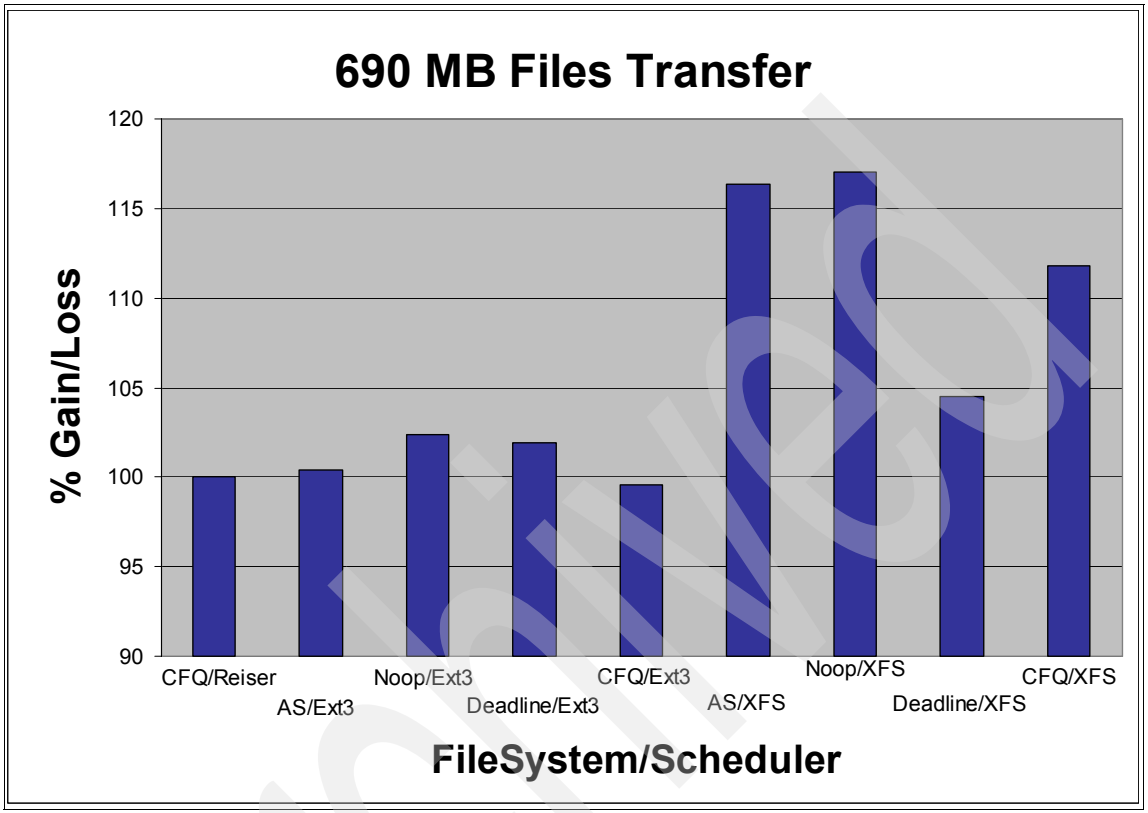


Figure 8-9 SUSE Linux Enterprise Server 690 MB file transfer times gains/loss

By analyzing the charts above, it becomes very clear that using XFS as the file system for a file server running SUSE Linux Enterprise Server on a System p servers is a very good choice. Within the tested environment, the usage of XFS yielded a gain of at least 10% for all file sizes when compared to the default Reiser file system, and when combined with the best scheduler for the specific file size, workload gains of over 20% were measured.

However, the task of choosing an I/O scheduler does not provide such an easy pick.The file size used in the workload resulted in different schedulers yielding the best results. However, for a workload where the file sizes range across the whole scope of this experiment, and based on the fact that we have a RAID 5 disk array controlled by a RAID SCSI adapter through a VIOS Server, the simple and very efficient NOOP I/IO scheduler proves to be a very good option.

This choice was validated when a run of the test scripts using the fast-Ethernet network combined with the virtual LAN of the OpenPower 720 server was done, where a larger number of remote clients were simulated. In this test, where a very mixed file size workload was used, gains of up to 40% in the transfer times were measured. See Figure 8-10 to check the performance gains and loss depending on the file size.

However, it is very important to mention that for environments where the server workload will be mainly composed of small files, the CFQ scheduler provided better performance gains than the NOOP scheduler, and it will likely be a better choice for servers submitted to this specific workload profile.

One additional benefit of selecting XFS and NOOP is the fact that this combination provided the lowest average CPU utilization during the file transfer, although it allocated about two times more memory for file caching at the server level than the CFQ scheduler. This detail should be taken into consideration when dealing with environments where memory and CPU resources available for the server are severely constrained.

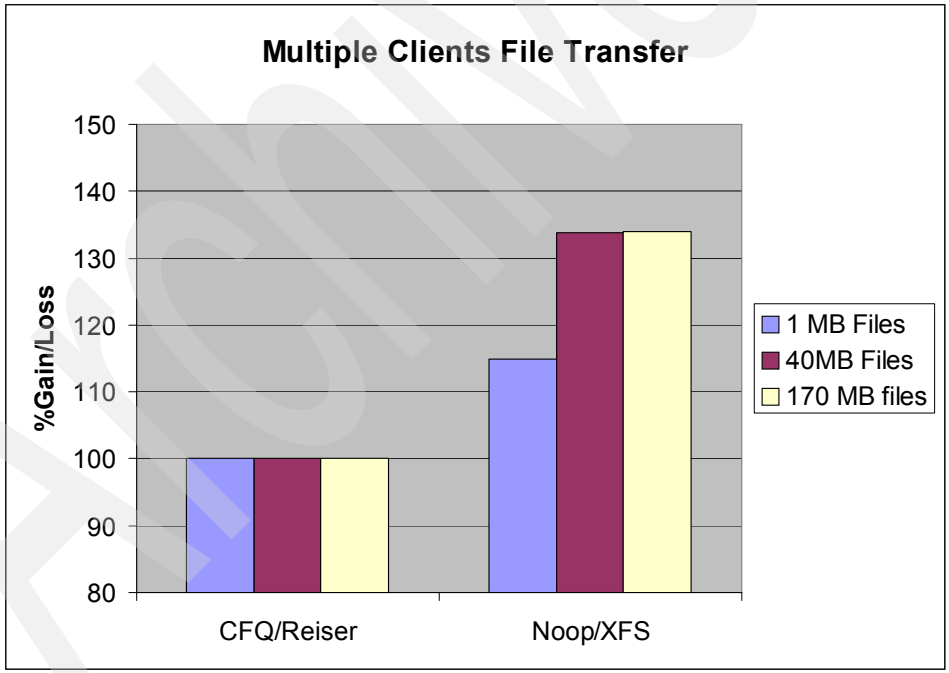


Figure 8-10 Comparison of the chosen file system and scheduler to the default options in a SUSE Linux Enterprise Server environment when the system bottleneck was forced to the disk

Red Hat Enterprise Linux measured performance gain/loss

Figure 8-11 shows the outcome of the 1 MB tests with Red Hat Enterprise Linux AS 4 update 4.

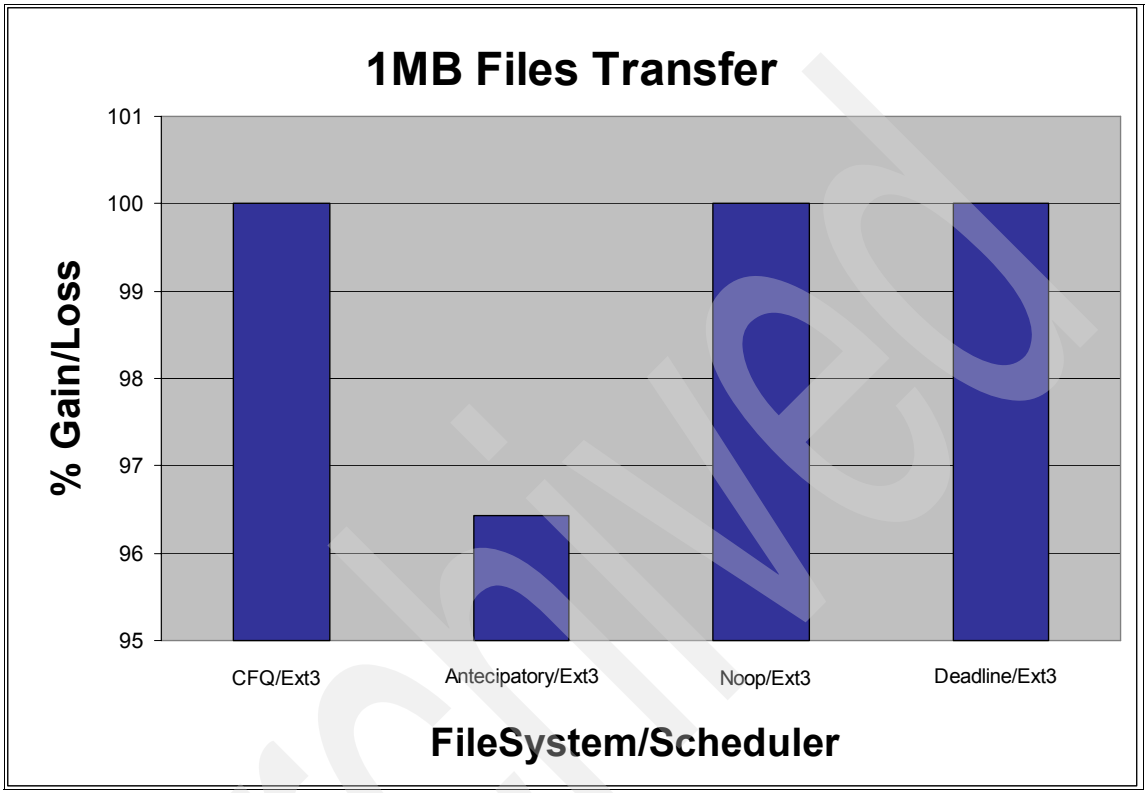


Figure 8-11 Red Hat 1 MB file transfer times gains/loss

Figure 8-12 shows the outcome of the 40 MB tests with Red Hat Enterprise Linux AS 4 update 4.

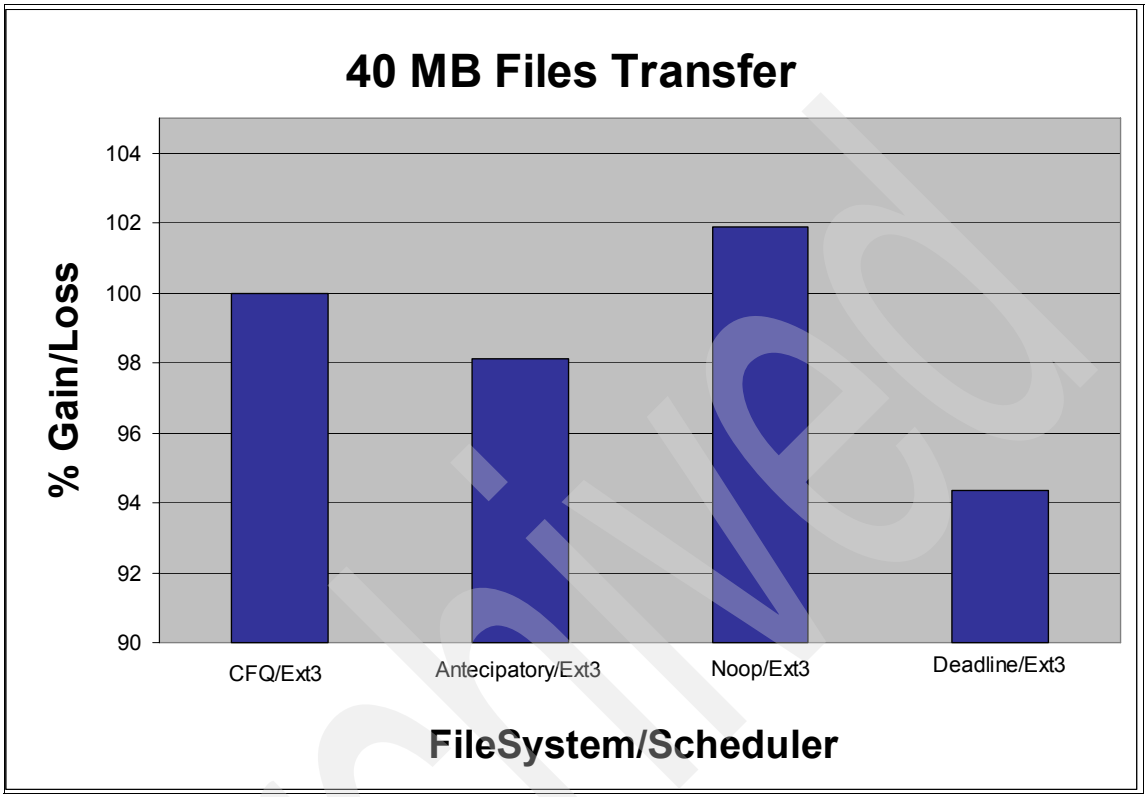


Figure 8-12 Red Hat 40 MB file transfer times gains/loss

Figure 8-13 shows the outcome of the 170 MB tests with Red Hat Enterprise Linux AS 4 update 4.

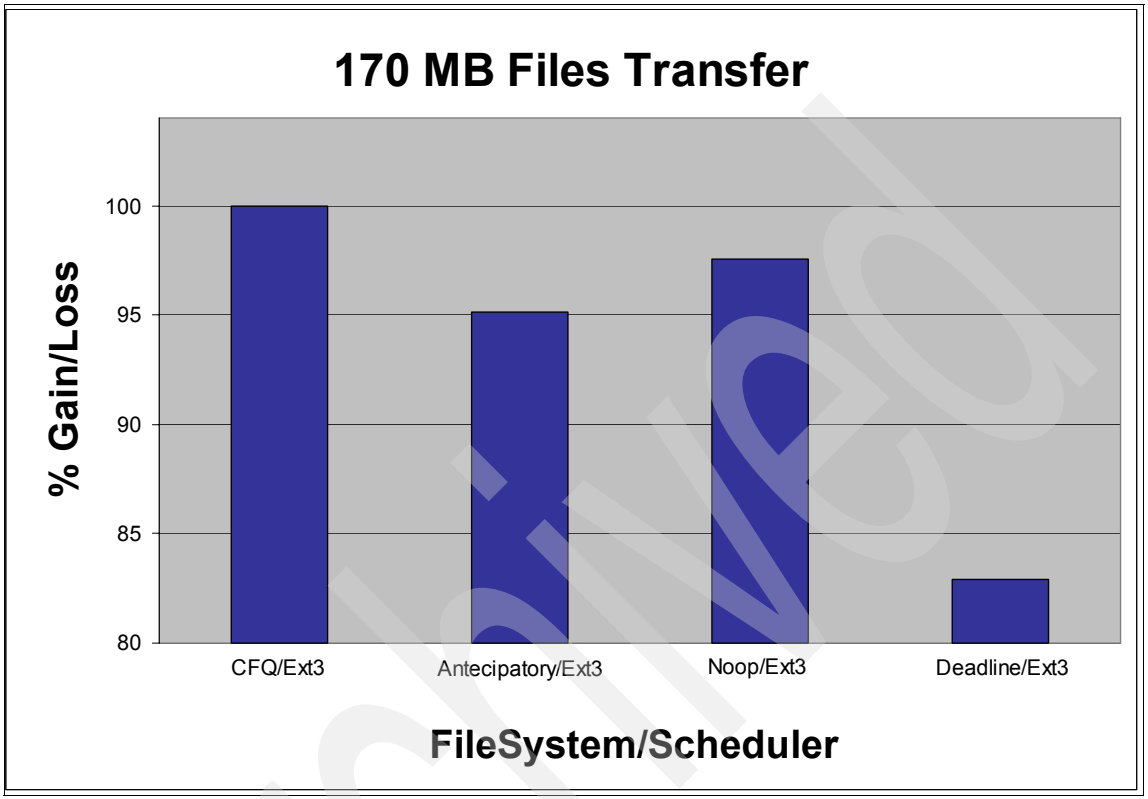


Figure 8-13 Red Hat 170 MB file transfer times gains/loss

Figure 8-14 shows the outcome of the 690 MB tests with Red Hat Enterprise Linux AS 4 update 4.

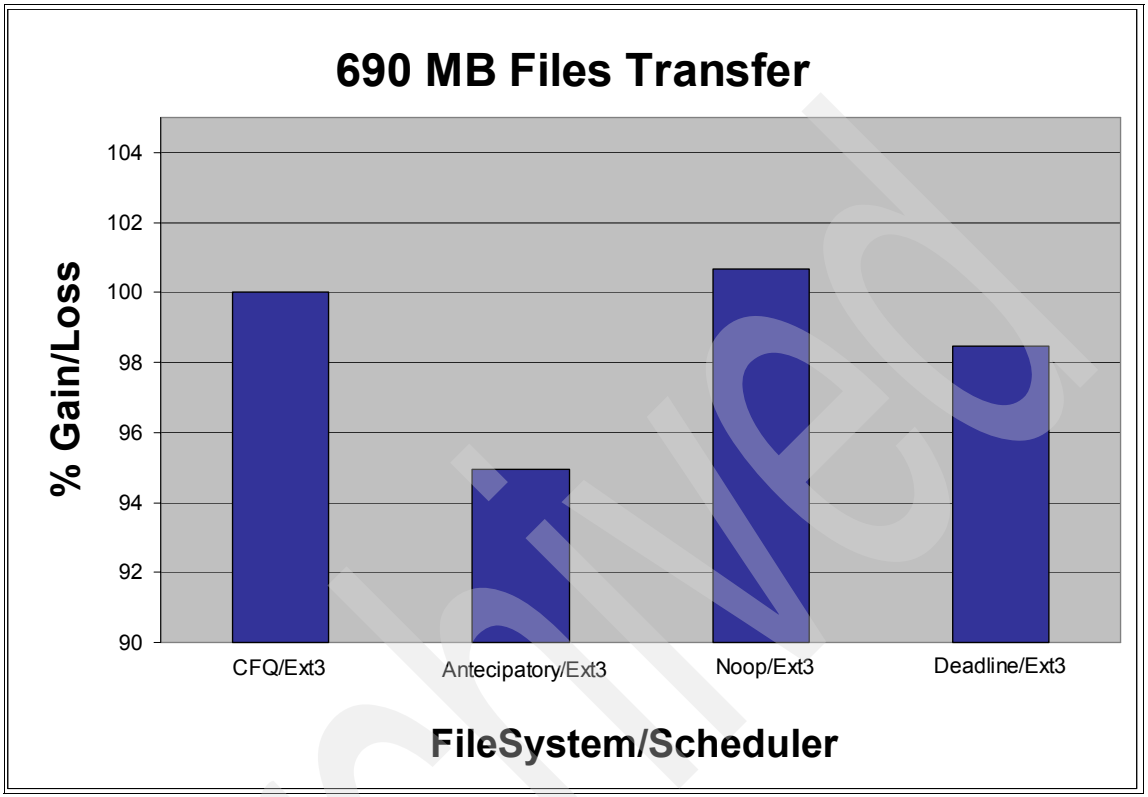


Figure 8-14 Red Hat 690 MB file transfer times gains/loss

Due to the fact that at the time of writing Red Hat Enterprise Linux AS 4 did not ship with support for XFS or ReiserFS, the ext3 file system is an automatic choice. Red Hat Enterprise Linux AS 4 comes with the four I/O schedulers already included in the kernel.

At the general availability (GA) time, Red Hat published an article providing a test scenario justifying the choice of ext3 file systems with CFQ as the default scheduler. The test results above indicate that on a System p environment, the choice is justifiable, as the anticipatory and deadline schedulers proved to be slightly slower (less than 5% for most file sizes tested and not enough to discard the results as simple statistical noise) while the NOOP scheduler provided results where the difference is less significant than the other schedulers. In systems that have a more advanced disk system, this difference could have tilted towards NOOP when combined with the ext3 file system.

Due to the fact that the results were very similar and the fact that the CFQ scheduler used less memory for cache than NOOP scheduler, choosing the CFQ in combination with *ext3* on the Red Hat Enterprise Linux AS 4 environment looks like the best option for further testing.

Conclusions

Here we discuss some of the conclusions we discovered.

I/O scheduler

As theorized, the nature of the disk system of the tested environment favored the simplicity of the NOOP scheduler in comparison to other schedules. The fact that a lot of intelligence is mounted into the disk controller and the VIOS helps environments where the simple first in first out (FIFO) architecture of NOOP thrives, especially when combined with a workload reach in large files.

However, for small files, the tests clearly demonstrated an advantage towards the CFQ scheduler.

Meanwhile, the anticipatory scheduler was not expected to perform well due to the paradigm used for its design; very limited disk I/O capabilities at the system, which was pretty distant from the environment tested.[1]

At last, the deadline scheduler proved to be a bad fit for this specific workload due to the write volumes involved. It underperformed considerably when compared to the other schedulers.

FIle system

The files performance impact clearly outshined the gains provided by the schedulers. In general, the schedulers by themselves did not provide a tremendous impact in terms of performance changes; however when combined with a different file system, different schedulers provided different levels of success.

Out of the charts in this section, it is very clear that XFS provides a significant gain for large files transfer as theorized. At the same time, ReiserFS does a very good job with the small files. These two options are enough to open the doors for a complete new study in terms of tuning each of the specific file systems.

[1] Even though the environment tested is no where near the performance of a high-end disk sub-system, it is still far ahead of the disk systems that could take advantage of the features in the anticipatory scheduler.

In order to provide a broader set of tuning options, from this point on, all relative performance comparisons will take into account the following:

- SUSE Linux Enterprise Server 10
  - Scheduler: NOOP
  - File System: XFS
- Red Hat Enterprise Linux AS 4
  - Scheduler: CFQ
  - File System: ext3

8.4.2 Tuning the scheduler

Here we discuss tuning the scheduler.

SUSE Linux Enterprise Server 10

The chosen I/O scheduler is very simple and no tunables variables are available at the sysfs (/sys/block/sda/queue/iosched). Therefore, there is no options for any I/O scheduler tuning for the SUSE Linux Enterprise Server environment.

Red Hat Enterprise Linux AS 4

Here, the CFQ scheduler available in Red Hat Enterprise Linux AS 4 allows for the tuning of two parameters:

- queued
- quantum

A series of tests was done changing the values for various combinations of the tunable parameters. Here we show the commands to change the value of the tunable parameters of the scheduler.[2]

- To change the queued value to X:

```
echo X >> /sys/block/sda/queue/iosched/queued
```

- To change the quantum value to X:

```
echo X >> /sys/block/sda/queue/iosched/quantum
```

[2] For readers with Kernel 2.4 experience, the eltune command was replace with entries at the sysfs virtual file system that allows you to tune the I/O scheduler.

The results of the tests with different combination of queued and quantum values can be found in Figure 8-15.

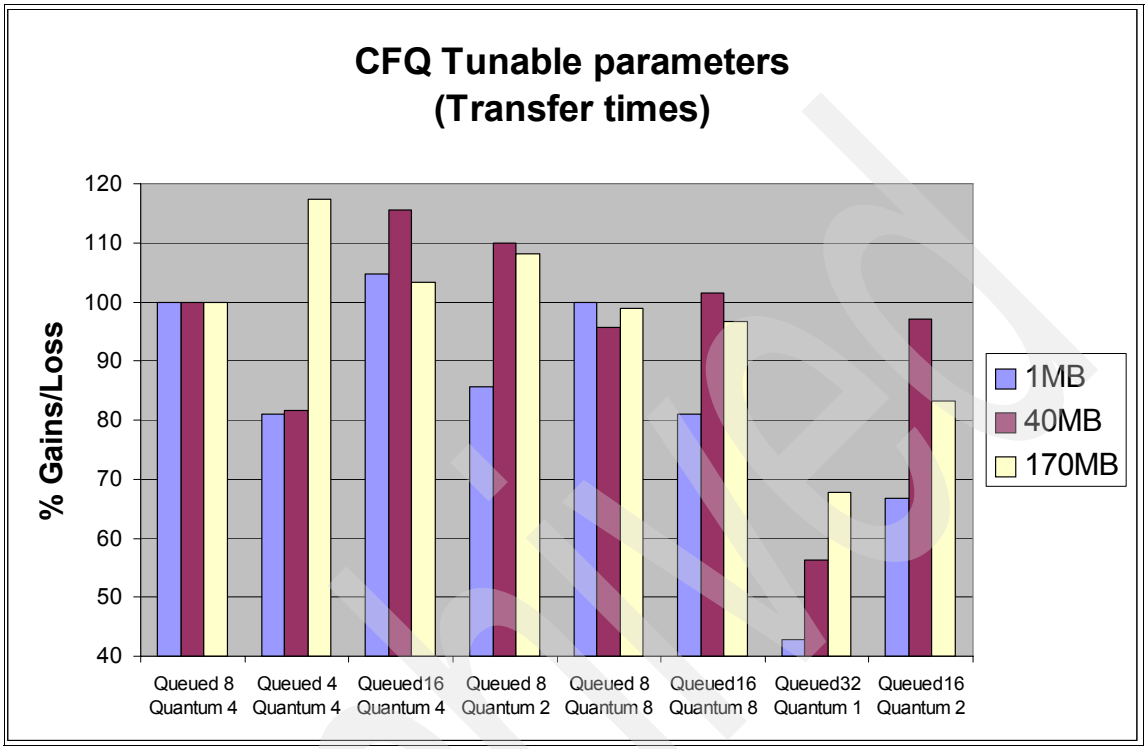


Figure 8-15 Comparison of the tunable CFQ I/O scheduler with the default options in a Red Hat environment when the system bottleneck was forced to the disk

For small files, the improvement is not statistically significant; however, for medium and larger files, the parameters can provide a gain of over 15% or more in the transfer times.

Note that the ratio of the parameters for larger files transfer seemed to be optimized one to one, while for medium and small files the one to four ratio seemed to provide the better output.

When a different workload is considered, different values may prove to yield better results, but when considering the test workload, a clear choice exists.

Another consideration to keep in mind is that as queues were added and the quantum decreased, a sensible increase in average CPU and memory utilization occurred. In a environment where the amount of CPU and memory available is limited, the tuning of these parameters may affect the overall performance considerably.

Conclusions

The NOOP scheduler does not provide any tunable parameters, therefore no work was done with the SUSE Linux Enterprise Server environment during this phase of tuning.

Meanwhile, Red Hat allowed for some tuning that provided some gains for medium and larger file sizes. With small files, no real significant gain could be found. However, a system administrator has to be careful about the tuning executed at the I/O scheduler level in order not to starve the CPU and memory resources.

With the results from this section, all further tuning will be made using the following values for the CFQ tunable parameters in the Red Hat Enterprise Linux environment:

- Red Hat Enterprise Linux AS 4
  - queued = 16
  - quantum = 4

8.4.3 Tuning the file system

Here we discuss tuning the file system.

SUSE Linux Enterprise Server

The chosen file system for the install, XFS, allows for a considerable amount of configuration tweaks. However, not all the options are a good choice.

The first parameter to be tested is the size of the journal, or log. At file system creation time, XFS, by default, creates a 22 MB log area. It is interesting to find out how a larger area affects the overall file system performance. In 7.7.3, “Tuning file synchronization” on page 197, there is a description on how to change the log buffer area for an XFS file system.

Another interesting tunable parameter is the number of log buffers used by the system. The log buffers are memory blocks that hold the log files in RAM. Each buffer consumes 32 KB of memory, which is not a very significant amount for the test environment, but it can be become an issue in a system with memory constraints. The default number for this parameter is two buffers, and higher values will be tested in order to understand how it affects the file system performance. Fortunately, this is a parameter that can be adjusted at mounting time. In 7.7.2, “Tuning file systems” on page 192, there is a description of how this change can be accomplished.

Additionally, the logging of access times can also present another opportunity for performance improvement, as most system administrators do not take advantage of this feature, and even the XFS file system creator, SGI, suggests turning it off to improve performance. Again, this is a parameter that can be changed at mount time. In 7.7.2, “Tuning file systems” on page 192, there is a description of how this change can be accomplished.

One last parameter that deserves to be mentioned is the XFS default feature of adding a flag to all unwritten files. Changing this parameter will theoretically cause a performance improvement; however, this is a safety feature, and as such, it should be left on.

Attention: To disable the flagging of unwritten files, add:

```
-d unwritten=0 to the mkfs -t xfs command
```

However it can cause considerable data loss during crashes, while providing minimal gains.

With all these considerations in mind, the same workload used in the previous tests was used and the changes can be seen in Figure 8-16.

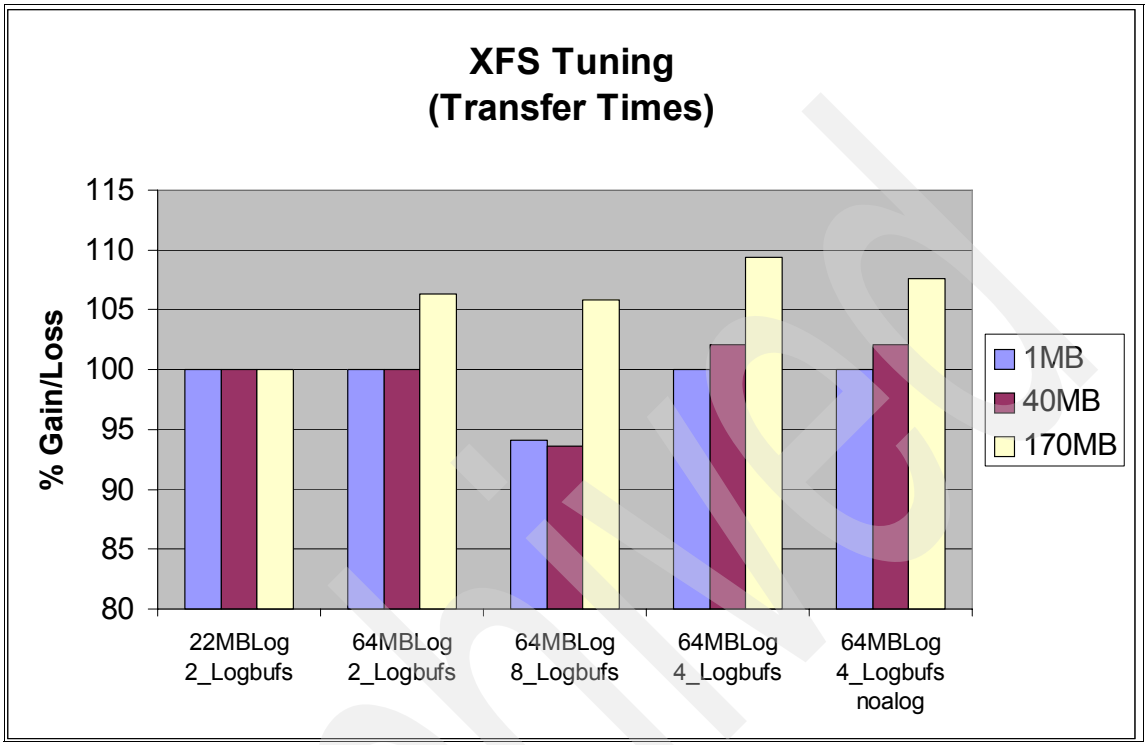


Figure 8-16 Comparison of the tunable XFS file system parameters in a SUSE Linux Enterprise Server environment when the system bottleneck was forced to the disk

When reviewing the results, gains can be seen for a 64 MB log size with the best results found when using four in memory log buffers. With the new parameters, gains of up to 10% in the performance can be seen for large files. However, for smaller files, the gains are not big enough to allow a positive identification that they are not a simple statistical anomaly.

A surprising result is the small degradation measured when the no access log option was included in the **`mount`** command. However, the difference is so small that it still fits into the variance of the test results and should be considered only a statistical anomaly. With this in mind, the access log will be set to off, as it will reduce the disk activity while yielding the same performance level.

Red Hat Enterprise Linux AS 4

Meanwhile, the Red Hat environment is using ext3, and it uses a completely different set of parameters for tuning.

Theory indicates that the biggest performance factor in ext3 tuning is the journaling method. Ext3 allows for 3 methods:

- Journal Data WriteBack
- Journal Data Ordered (Default Method)
- Journal Data

Details about how the journaling method works can be found in section 7.7.3, “Tuning file synchronization” on page 197.

Tip: In order to change the journaling method just run the **`tune2fs`** command for the proper file system with the parameter:

```
-o journal_data_writeback
```

Another factor to be tested is disabling of the access log function at the file system level. Although no real performance gain should be expected due to the previous result, the general impact is positive, with no documented case where this affects an application.[3]

With all these considerations in mind, the same workload used in the previous tests was used and the changes can be seen in Figure 8-17 on page 237.

[3] It is theorized that some makefiles may use the information, but no reference of a real example could be found at this time.

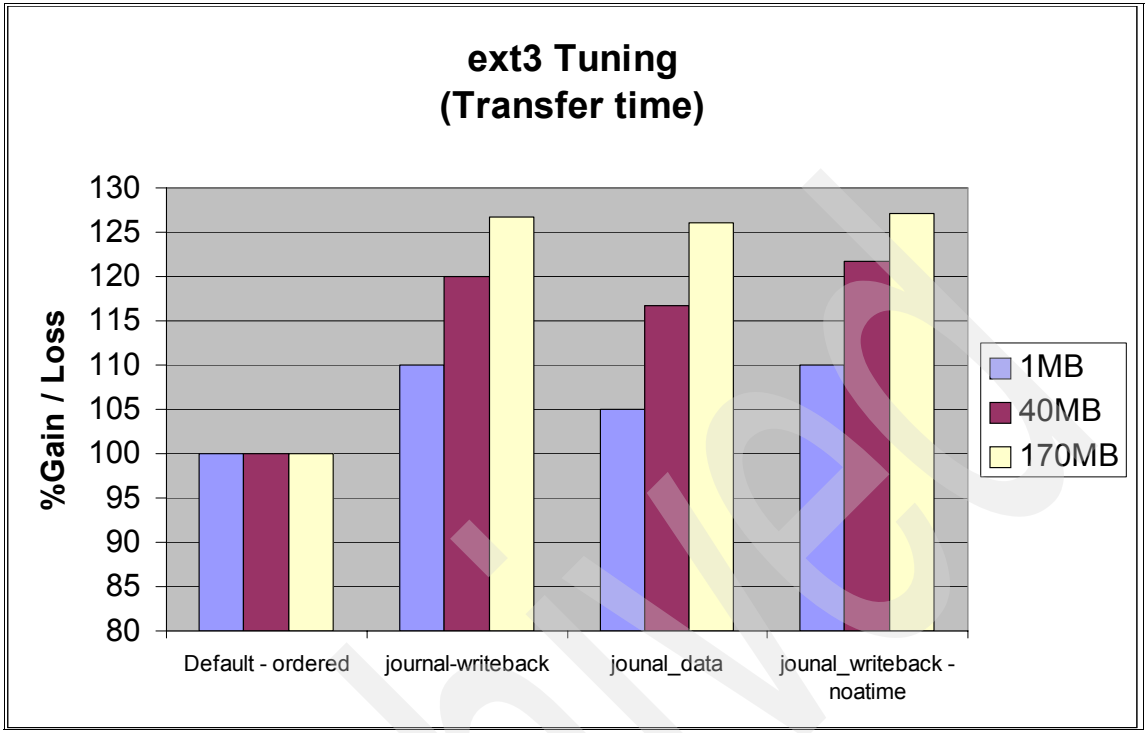


Figure 8-17 Comparison of the tunable ext3 file system parameters in a Red Hat Enterprise Linux environment when the system bottleneck was forced to the disk

The change in the journaling method provided a big improvement in the specific workload. It yielded gains that ranged from 10% to over 25%, but the system administrator has to be aware that the change of the journal logging algorithm affects the way data and metadata are updated, so a considerable data loss may occur.

Also, the differences seen after disabling the access logs are not big enough to be ruled as anything but a statistical anomaly, but due to the positive impact in I/O, it will also be turned off during further tests.

Conclusions

Tuning how a journaled file system handles the journal logs can provide considerable performance gains. However, a system administrator must be aware of the consequences, as both XFS and ext3 defaults are set to provide as much security as possible by minimizing the delay of the writes to the files and the logs.

Additionally, other parameters that affect the tuning do exist, but they would require a considerable larger amount of time to properly test, as they in general require changes to be made at file system creation time, which would make any further testing of other sub-systems not viable for this study. Two examples that are very well documented are the journal log placement in a different disk and the selection of a block size that properly fits the workload.

However, a system administrator is highly encouraged to try out the tuning recommendations described in 7.7.2, “Tuning file systems” on page 192 to properly set up the file systems with the parameters that will provide the best compromise of performance and security for his or her specific workload.

Moreover, proper planning of the file system structure for the server has to be done in advance, as many of the parameters are adjustable only through the command line and at file system creation time.

A very interesting fact is that, unlike the process of choosing and tuning a scheduler, the processor or the memory did not suffered any significant change in utilization due to any of the parameters changes executed at the file system level.

Finally, the final changes for the file systems will be:

- SUSE Linux Enterprise Server 10
  - 64 MB log area for the file system.
  - Use 4 log buffers in memory.
  - Disable the access logging.
- Red Hat Enterprise Linux AS 4
  - Use journal data writeback mode.
  - Disable the access logging.

8.4.4 Tuning the network interface

Linux provides a long list of options to tune the TCP/IP stack and the network parameters. In 7.8, “Tuning the network subsystem” on page 200, there is a very extensive discussion of the tunable parameters available.

For testing purposes, the various parameters were grouped into five different main functions:

Default	Linux default parameters after boot.
Unwanted Services	A series of TCP and UDP services that are related to functions usually not provided by a file and printer server.
TCP Buffers	Parameters related to how Linux uses the memory for TCP traffic buffers.

TCP Options and Connections
: Parameters related to how the TCP stack handles TCP windows and acknowledges as well as timeouts for connections reset.

IP Fragmentation
: Parameters related to memory allocated to assemble fragmented datagrams.

The test was executed in the same environment described in 8.3, “The file server testing environment” on page 214, but instead of testing the effects of each set of parameters individually, the test was done in an incremental way. Starting with the default values, we tuned the Unwanted services, then the TCP Buffers, and so on.

It is very important to keep in mind that in a file serving production environment, the network is the subsystem with the highest possibility of being the initial bottleneck of an untuned Linux box. In the test environment, it was very clear that the network was the main bottleneck. During the tests, while stressing the network subsystem, the CPU, memory, and disk average utilizations were never at a significant level, and never peaking to a utilization level of concern.

SUSE Linux Enterprise Server 10

In Figure 8-18, the gains measured after tuning the network system parameters groups is shown.

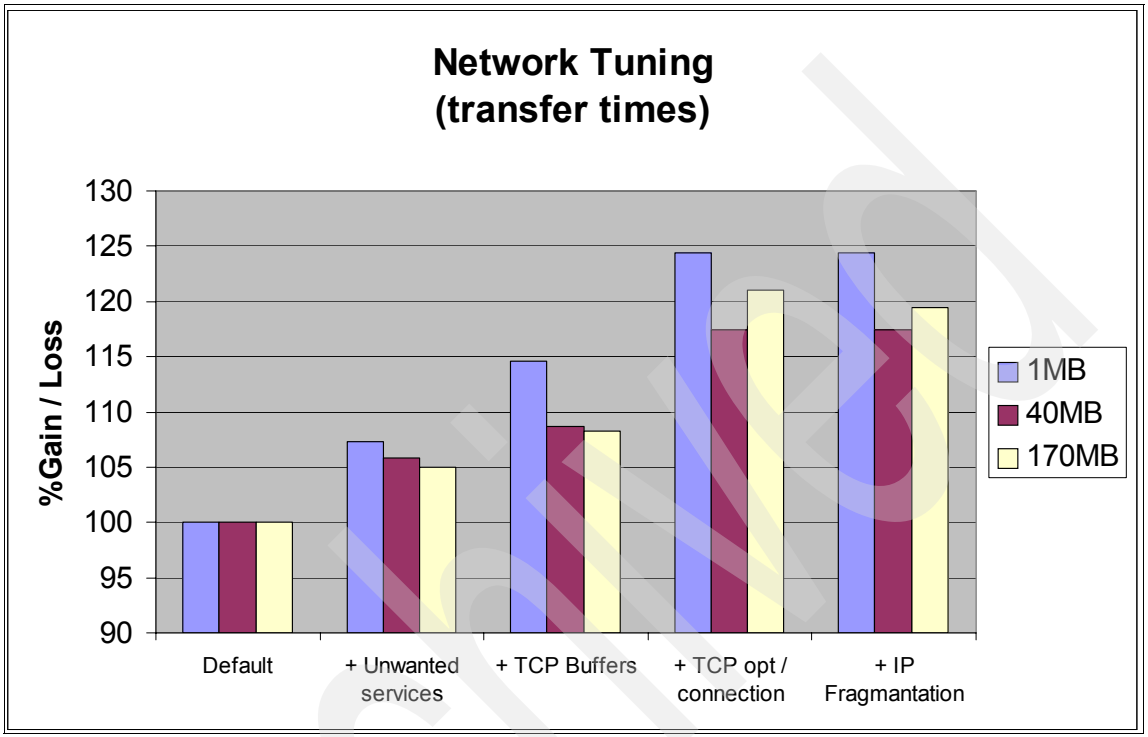


Figure 8-18 Comparison of the tunable network parameters in a SUSE Linux Enterprise Server environment when the system bottleneck was forced to the network

The results indicated that the effect of tuning the IP fragmentation buffers provides virtually no change to the transfer times, as the differences are not big enough to be discarded as statistical noise.

Additionally, it is interesting to know how the bulk of the performance gains come from the tuning of the TCP options and connection management parameters.

Red Hat Enterprise Linux AS 4

In Figure 8-19, the gains measured after tuning the network system parameters groups is shown.

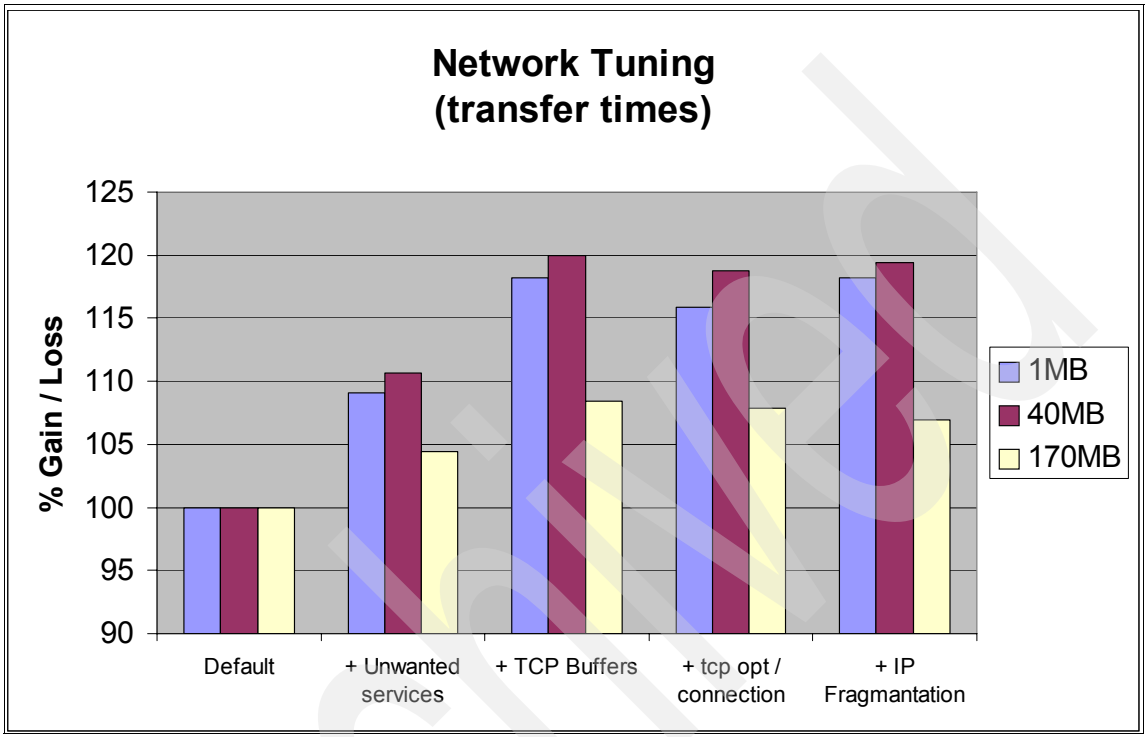


Figure 8-19 Comparison of the tunable network parameters in a Red Hat environment when the system bottleneck was forced to the network

Again, the results indicated that the effect of tuning the IP fragmentation buffers provides virtually no change to the transfer times, as the differences are not big enough to be discarded as statistical noise.

Unlike the SUSE Linux Enterprise Server tuning, the bulk of the performance gains are realized by the TCP Buffers tuning and not by the options and connections management. From that point on, the differences in results are so minimal that any effects seems to be caused by statistical noise, and not by an actual change in the performance characteristics of the networking subsystem.

Additionally, in the Red Hat Enterprise Linux environment, most gains were achieved with medium files instead of small files, although the gains difference is not that significant, as it still is within the statistical noise range.

Conclusions

Tuning the network parameters provide, for both distributions, gains of up to *25%* in the transfer times.

It is very important that a system administrator is aware that the tests were executed in a environment where an extremely fast network was dedicated for the tests, and no other network usage was involved. This is not typical for real client environments, and the network subsystem gains can be easily masked by the network usage by other services and applications. A system administrator should be aware of the network environment where the actual production system will be placed, as this allows for better tuning of the Linux networking subsystem to yield the best results for each individual environment.

The tests actually provided two very interesting results. The first was the fact that the IP fragmentation buffers increase did not provide any meaningful gains. This indicates that for the test environment the Linux default values are adequate.

The second interesting result was the set of parameters that provided the bulk of the performance gains for each of the distribution. In SUSE Linux Enterprise Server, the bulk of the gains came from the tuning of the TCP options and connections parameters, while in Red Hat Enterprise Linux the bulk of the gains came from tuning the TCP Buffers.

Such a result may seen surprising, but it is important to remember that both distributions use different default values for the TCP Buffers parameters in their PPC versions of Linux.

It is also very interesting that tuning the network provided the first subsystem where the results showed significantly better gains when transferring small files instead of large files, especially in the Red Hat Enterprise Linux environment.

With these considerations in mind, all parameters sets will be modified to the tested values, with the exception of the IP fragmentation buffers, as they proved to not add any significant change to the results.

Therefore, the final modifications to tune the network parameters will be:

- SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4
  - Unwanted services
    - net.ipv4.conf.eth0.secure_redirects = 1
    - net.ipv4.conf.lo.secure_redirects = 1
    - net.ipv4.conf.default.secure_redirects = 1
    - net.ipv4.conf.all.secure_redirects = 1
    - net.ipv4.conf.eth0.accept_redirects = 0
    - net.ipv4.conf.lo.accept_redirects = 0
    - net.ipv4.conf.default.accept_redirects = 0
    - net.ipv4.conf.all.accept_redirects = 0
    - net.ipv4.conf.eth0.send_redirects = 0
    - net.ipv4.conf.lo.send_redirects = 0
    - net.ipv4.conf.default.send_redirects = 0
    - net.ipv4.conf.all.send_redirects = 0
    - net.ipv4.icmp_echo_ignore_broadcasts = 1
    - net.ipv4.icmp_ignore_bogus_error_responses = 1
  - TCP Buffers
    - net.ipv4.tcp_rmem = 32768 436600 873200
    - net.ipv4.tcp_wmem = 8096 436600 873200
    - net.ipv4.tcp_mem = 13145728 14194304 16291456
  - TCP Options and connection management
    - net.ipv4.tcp_window_scaling = 0
    - net.ipv4.tcp_sack = 0
    - net.ipv4.tcp_dsack = 0
    - net.ipv4.tcp_fack = 0
    - net.ipv4.tcp_max_syn_backlog = 2048
    - net.ipv4.tcp_synack_retries = 2
    - net.ipv4.tcp_retries2 = 10
    - net.ipv4.tcp_keepalive_time = 3600
    - net.ipv4.tcp_keepalive_intvl = 50
    - net.ipv4.tcp_keepalive_probes = 5

Moreover, due to the nature of file transfers, especially when considering large files, the standard 1500 byte MTU of Ethernet networks can be a critical factor as a performance deterrent. Because disk data blocks are in general considerably larger than the standard 1500 bytes MTU of a Ethernet and gigabit Ethernet frame, many CPU cycles are wasted handling IP fragmentation and header creation.

A solution for this problem in gigabit Ethernet networks is the use of jumbo frames all the way from the server to the client.

Jumbo frames allows Ethernet frames of up to 9000 bytes, which have two very important effects when considering file servers.

The first effect is the capability of the server to send a full datagram from the file serving protocols with a single packet, so no fragmentation will happen at the Ethernet level (layer 1 and 2 of the OSI model).

The second effect is a consequence of the reduced number of packets sent through network. With less packets to process, the server is able to reduce the CPU utilization considerably, therefore freeing cycles to handle other operations done by the server.

If more information about jumbo frames is needed in order to understand how it affects a system performance, the following articles available on the internet can prove very useful:

http://sd.wareonearth.com/~phil/jumbo.html
http://www.small-tree.com/resources/pdfs/jumbo_white_paper.pdf

8.4.5 Tuning CPU

In file serving, a common misconception is that CPU capacity is important. CPU rarely is a source of performance bottlenecks for file servers. However, the nature of the workload is that it tends to increase considerably the ratio of CPU cycles used by the userspace (user) and the system calls (sys), especially when the transfer block sizes are small. Interestingly enough, there was virtually no I/O Wait (wait) when performing the CPU stress tests as the files were cached after the first access, as the system had plenty of memory.

However, when a bottleneck in CPU subsystem exists in a file server, it is easily detectable. As shown in Figure 8-20 and Figure 8-21 on page 246, the transfer times increase exponentially as the CPU resources come into the saturation range.

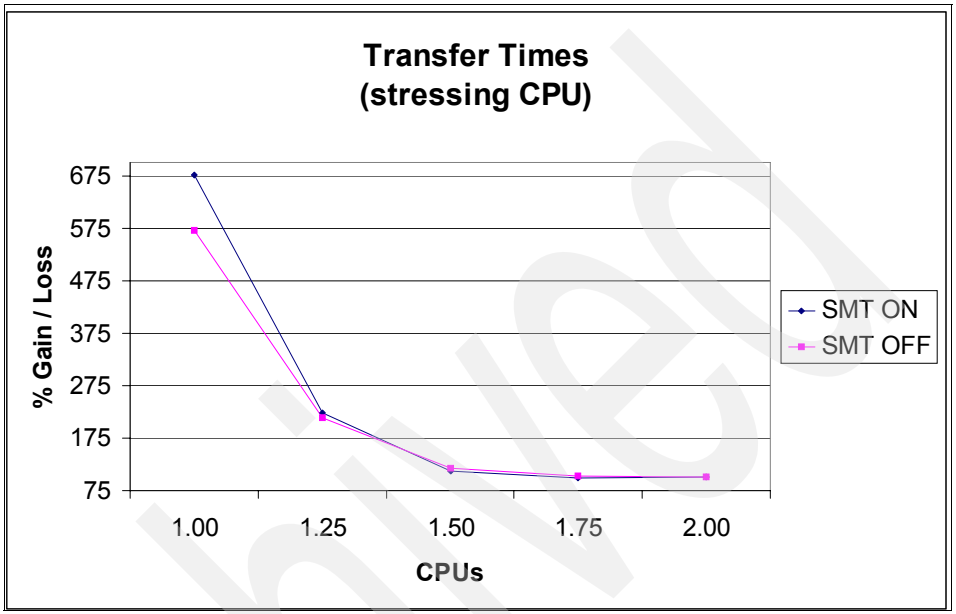


Figure 8-20 Effects of CPU starvation in the transfer times for a SUSE Linux Enterprise Server file server

A very interesting result that can be seen in Figure 8-20 and Figure 8-21 on page 246 is that if a server CPU resources are not saturated, adding extra CPUs will provide no significant gains in the transfer times. In the specific case of the workload used in this study, no significant gains were measured when the file server was configured with more then 1.75 CPUs, and a slight performance degradation could be measured starting at 1.5 CPUs. This degradation compounded very quickly to the point where the transfer rates increased by more than three times the original measured values.

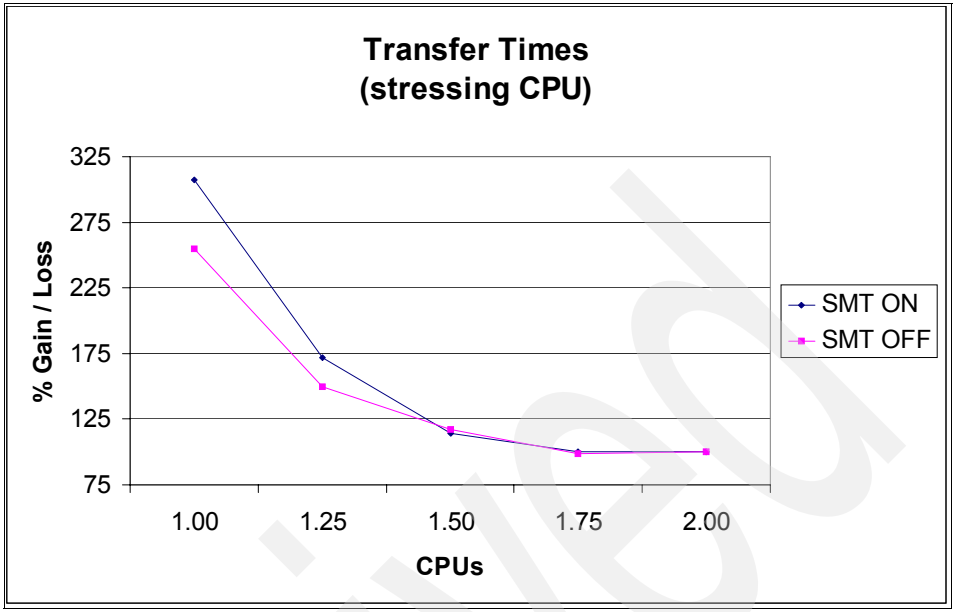


Figure 8-21 Effects of CPU starvation in the transfer times for a Red Hat file server

However, there are ways to mitigate the point where the exponential curve starts the fast degradation.

The easiest way to free CPU cycles to mitigate the performance problems is to disable unnecessary services, as described in 8.4.7, “Disabling daemons” on page 251. Without these services using CPU cycles, the file serving daemons will have more cycles available to process service requests.

Another way to reduce CPU consumption is to use *jumbo frames*, if they are supported by the network throughout all systems accessing the server. As described in 8.4.4, “Tuning the network interface” on page 238, jumbo frames will allow the TCP stack to handle less incoming and outgoing network packets, therefore freeing considerable resources used for system calls related to processing these packets.

Additionally, tuning the kernel parameters listed in 7.4, “Kernel parameters” on page 170 can provide significant results in environments where a mixed workload is put into the system, but this is not the case for this study.

Another CPU related feature that affects performance is the POWER5 Simultaneous Multithreading Technology (SMT). A good overview of how SMT works can be found in 2.3.1, “Processor subsystem” on page 30.

In Figure 8-22 and Figure 8-23 on page 248, there is a ratio comparison of the transfer times for the same amount of CPU resources available for the server when SMT is enabled (ON) in comparison to the same test executed with SMT disabled (off). The value of SMT disabled would be at 100 for each data point. If the new datapoint is a transfer time of 92, it means that with SMT on the test was 8% faster.

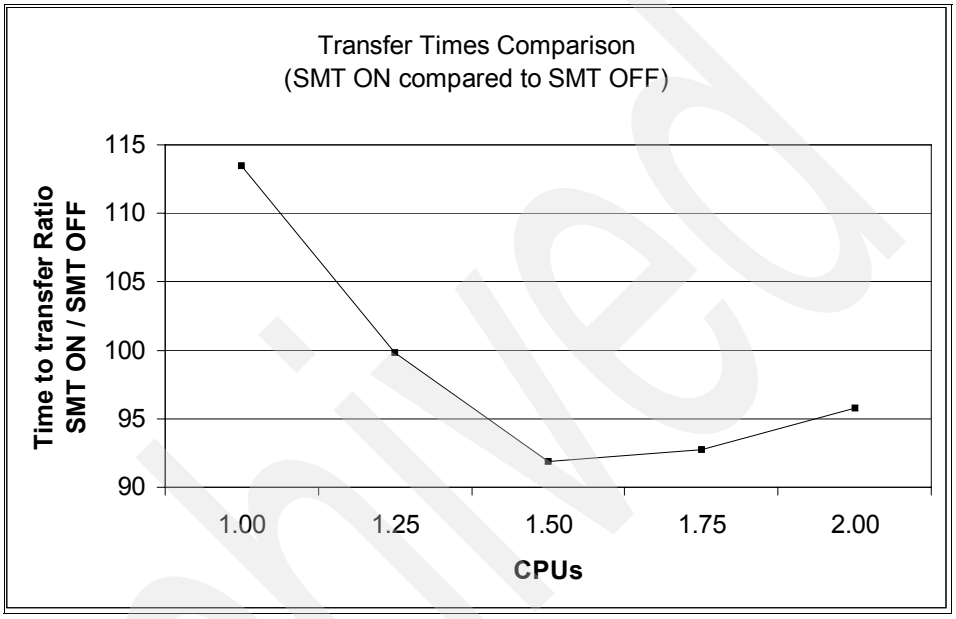


Figure 8-22 Transfer times ratio with SMT ON compared to SMT OFF in a SUSE Linux Enterprise Server environment

It is interesting that as the CPU resources starts to starve, if SMT is enabled, the transfer times actually increases considerably. As soon as the system has enough resources to get out of extreme starvation, SMT starts to provide noticeable gains due to better parallelism of the instructions lined up to be executed. Eventually, when the system gets out of any form of CPU starvation, the advantages of executing more within a single CPU cycle starts to diminish and the gains start to shrink.

In the end, system administrators have to keep careful track of where in the CPU starvation curve their managed systems are located, and then make the decision of whether they want to have SMT on when they swing from one side of the curve to the other.

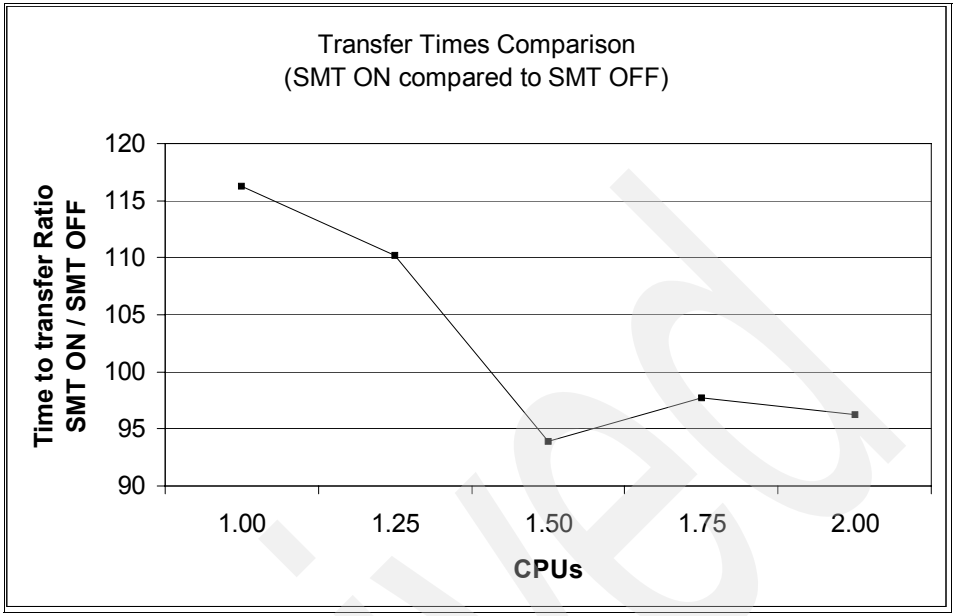


Figure 8-23 Transfer times ratio with SMT ON compared to SMT OFF in a Red Hat Enterprise Linux environment

However, if a system does not have the CPU as its primary bottleneck, then SMT is able to provide beneficial results to file servers that in the environment described in 8.3, “The file server testing environment” on page 214 proved to be up to 8%.

One last function that has to be considered when tuning the CPU subsystem of file servers is the System p capacity to create virtual processors out of fractions of physical processors.

A bigger number of CPUs can be beneficial for processes that are not designed with good multithreading capabilities and tend to take ownership of a CPU. This way, the system will be able to context switch these processes out without any need to change the software. However, for processes properly designed to take advantage of multithreading and CPU “mobility”, a large number of CPUs can cause a performance degradation due to the sheer number of unnecessary context switches necessary to manage all the virtual processors.

For the sake of time and consistency, all tests in this study were executed with two virtual processors in the server, but systems administrators are encouraged to test how the number of virtual processors assigned to a partition will affect the performance results.

Conclusions

CPU in general is not a factor in a properly designed file server, but it becomes very clear when a CPU bottleneck exists.

At the same time, the advanced POWER5 features like SMIT and virtual processors, when combined with Linux, can provide performance benefits, but the CPU subsystem has to be properly monitored in order to avoid the possibility of a backfire when performance enablers turn into performance deterrents.

In the tested environment, the final results for the tuning yielded:

- SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4
  - Simultaneous Multithreading Technology (SMT)
  - The use of two VP processors for the server when system has two or less physical CPUs allocated

8.4.6 Tuning memory

Insufficient memory might often be the reason behind poor server performance. As the amount of memory that running programs and their data uses approaches the total available physical memory that is installed on the machine, a server's virtual memory handler increases the amount of data that is paged in and out of memory, to and from the paging area on disk, with a disastrous effect on performance. Fortunately, the memory subsystem is usually one of the easiest areas of the entire system to upgrade and measure utilization levels.

Over the years, memory capacity demands have increased because server operating systems and server application code have grown. Additionally, user content has evolved from simple character-based data to more expansive rich media, such as audio and video. Trends indicate an increasing demand for server memory capacity well into the future.

When comparing the speed of CPU cache, main memory, and online disk storage media, we can see that cache is the fastest but also the most expensive per megabyte, while disk is the cheapest per megabyte, but orders of magnitude slower than cache memory. The use of main memory is a good compromise between speed and price. The L1, L2, and L3 caches bridge the performance gap between processor and main memory. Meanwhile, main memory can be used to bridge the performance gap between the caches and disk.

Although Linux allows for tremendous control of the swapping parameters, System p servers, especially when running in a virtualized environment, are strongly encouraged to work in a configuration where no memory swap to disk occurs. Therefore, in a properly tuned file server, we expect that the operating

system will not be doing any paging, and still have enough memory to provide disk cache for the most recently accessed files.

Tests of the file server indicated that for each user, up to 8 MB of memory can be used to manage the file transfer transactions. Therefore, a system administrator has to make sure that the main memory is able to handle the following:

- Linux kernel
- Active daemons (the basic daemons for samba, *smbd* and *nmbd*, can easily consume 20 MB of memory, while the *NFS related* daemons can consume 15 MB or more)
- The number of active simultaneous users (8 MB per user is a good conservative estimate; however, during the tests for this book, some combinations of parameters took the memory use up to 20 MB per user during the heaviest workloads tested)
- Memory for disk caching

With these considerations, we recommend at least 512 MB as *a starting point* for a file server if the server is not going to be heavily utilized. For servers with heavier workloads, only adequate testing can provide the optimum amount of memory required.

Tip: During the disk, network, and CPU stress tests of the test environment described in 8.3, "The file server testing environment" on page 214, the memory utilization by the server consistently was above 1 GB and sometimes went over 2 GB.

If a system administrator is really determined to improve the system memory utilization, one of the most effective method to do so is through disabling daemons that are not related to the main workload of the server. In 8.4.7, "Disabling daemons" on page 251, you can find information about some of the default daemons that can be stopped in a file serving environment.

Conclusions

The bottom line is that swapping should be avoided at all costs if performance is a key factor in a file server as, usually, the disk system is already heavily taxed by the normal workload of the server.

- SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux AS 4
  - Use 4 GB or RAM for the server, as this amount proved to be enough to keep the disk cache at excellent levels and no paging occurred

8.4.7 Disabling daemons

As described in 7.1, “Disabling daemons” on page 160, there are several daemons that are started automatically in a default Linux install. Not all of these daemons are used in a system that is acting as a file server.

Due to the test environment described in 8.3, “The file server testing environment” on page 214, the results will not be significant; however, in a environment where CPU and or memory utilization is very high, the extra resources freed after disabling these daemons may provide a significant performance boost.

In a SUSE Linux Enterprise Server 10 environment, the following daemons can be stopped when the system is acting as a file server:

- alasound
- postfix
- powersaved
- splash_early
- splash
- fbset
- xdm
- xfs

Additionally, if the system is not acting as a printer server, the cups daemon can be disabled as well. The same applies to the xinetd daemon if none of the services it provides is in use (telnet, time, and klogin are examples of services provided by xinetd)

In a Red Hat Enterprise Linux AS 4 environment, the following daemons can be stopped when the system is acting as a file server:

- autofs
- sendmail
- isdn
- pcmcia
- xdm
- xfs

Additionally, if the system is not acting as a print server, the cups, cups-conf-daemon, cups-lpd, and the hpoj daemons can be disabled as well. The same applies to the xinetd daemon if none of the services it provides is in use (telnet, time, and klogin are examples of services provided by xinetd).

Tip: Independent of the flavor used, if the file server is providing SMB/CIFS services only, then the various NFS daemons can be stopped as well.

8.4.8 Tuning Samba

Samba has a set of parameters that can be set up to allow improved performance in a file serving environment. We now discuss some of the main settings to optimize the performance of a Samba server. All these settings can be found in the /etc/samba/smb.conf file.

oplocks

This option lets SMB clients cache files locally. If a server grants an oplock (opportunistic lock), then the client is free to assume that it is the only one accessing the file and it will aggressively cache file data. This can provide big performance benefits. oplocks is enabled by default and should only be disabled when dealing with unreliable network connections. It is also possible to enable or disable oplocks on a by-share basis. The default value for enabling oplocks is:

```
oplocks = yes (default)
```

level2 oplocks

level2 oplocks (read-only oplock) allows Windows NT® clients that have an oplock on a file to downgrade from a read/write oplock to a read-only oplock. This will happen if another client accesses the same file on the Samba server. This allows all openers of the file to cache the file for read-ahead only and increases performance for many accesses of files that are not commonly written (such as application files).

Once one of the clients that have a read-only oplock writes to the file, all clients are notified and told to break their oplocks to “none” and delete any read-ahead caches. We recommend that this parameter be turned on to speed access to shared executables. The default value for enabling level2 oplocks is:

```
level2 oplocks = yes (default)
```

Socket options

There are a number of socket options that can greatly affect the performance of a file server like Samba. Getting the socket options right can make a big difference to the performance of the Samba server. The correct settings are very dependent on the network environment.

Socket options are targeted at the TCP networking layer. Within Samba, these options can be specified on the command line with the -o option or with the sock options parameter within the Samba configuration file.

By default, both Red Hat Enterprise Linux and SUSE Linux Enterprise Server have already set these options for a good performance value when installed in a local area network (LAN):

```
socket options = TCP_NODELAY (default)
```

Tests indicate that in read intensive environments, the gains can be of up to 50%; unfortunately, that is not the case for the test environment used for this study. As a default parameter in both distributions, it is already taken into consideration when reviewing the results reported above.

On the other hand, in installs where the files will be accessed through a wide area network (WAN), the following parameters can improve the performance:

```
socket options = ITPOS_THROUGHPUT TCP_NODELAY
```

Log level / Debug level

Another area that can dramatically decrease performance is the log level option (also known as debug level). If the system administrator needs to debug a problem, then the logs will provide valuable information to troubleshoot problems. The amount of information stored is based on the value of the variable log level. The default value is zero, which means no logging. The higher the value, the more detailed will be the logging, and the harder will be the performance impact. To maximize performance in production environments, make sure that this variable is set to zero if it is present in the smb.conf file.

```
log level = 0 (default)
```

Raw reads and writes

The last parameter that can improve performance is the raw reads/writes. This parameter allows clients to transfer larger chunks of data as a single packet. However, some clients do not support this feature and may force the server administrator to disable this feature. The default for these parameters is yes if it is not included in the smb.conf file.

```
read raw = yes (default)
write raw = yes (default)
```

Conclusions

Both SUSE Linux Enterprise Server and Red Hat Enterprise Linux have done a good job at tuning the performance related parameters of the Samba configuration file (/etc/samba/smb.conf) and most of the parameters that can affect performance have the default value tuned for performance.

In some specific cases, a system administrator will be forced to change the parameters to improve the performance of the Samba server for specific scenarios. A good source of information to track is the man page for the smb.conf file. The page can be accessed through the command:

```
man smb.conf
```

8.4.9 Tuning Network File System

Network File System (NFS) has a set of parameters that can be set up to allow, both at the server and the client, testing that can provide improved performance in a file serving environment. We now discuss some of the main settings to optimize the performance of an NFS.

Block size

When mounting an exported NFS in a client, the options rsize and wsize specify the size of the chunks of data (block) that the client and the server will pass back and forth. Although the default values vary depending on which version of NFS is being used, the most common value is 4 KB (4096 bytes). However, these are not the maximum values for these parameters. While NFSv2 allows for a maximum of 8 KB, NFSv3 and NFSv4 allows blocks of up to 32 KB.

Depending on the specific combination of hardware, the default can be too small or too big. However, in general, if a client is able to handle a bigger block size, the performance should increase. However, this tuning is completely dependent on the client side capabilities and performance tuning has to be done on a case by case basis.

It is important to keep in mind that if larger blocks are used, jumbo frames become a very important performance factor. A jumbo frame enabled network is able to handle larger packets without the need to fragment IP packets while transmitting blocks of data. More information about jumbo frames can be found in 8.4.4, “Tuning the network interface” on page 238.

In order to change the block size for a NFS export at a client at mount time, include the following parameters in the **mount** command:

```
-o rsize=x wsize=y
```

where x and y are the desired block sizes

NFS over TCP or UDP

Current NFS servers and clients are able to transfer the NFS protocol over either TCP or UDP.

The default UDP protocol, with its stateless connection characteristics, will provide the best performance on stable networks that have similar performance characteristics. Additionally, clients do not hang when a server crashes.

However, TCP offers considerable advantages when the network is not stable, and a measurable amount of packets are lost, as TCP allows for single packet retransmissions instead of an entire RPC request transmission. Additionally, TCP connections are able to handle network speeds differences better than UDP.

In most cases where a NFS export will be mounted within a local area network (LAN), UDP will be the best choice from the performance point of view. But in situations where a NFS export will be mounted over a wide area network (WAN), or the networks involved have a significant percentage of lost packages, TCP is very likely to provide better performance.

In order to change the protocol to TCP, include the following option at mount time:

```
-o tcp
```

Timeout and retransmissions

When using UDP as the protocol, NFS allows a system administrator to fine-tune the behavior of NFS connections when dealing with client timeouts, dropped packets, or both. The -o timeo option designates the amount of time, in tenths of a second, that a client will take to reissue a request to server when no response is received. On the other hand, the - restrans parameter indicates how many times a client will retry the same request before it displays the `server not responding` message.

Again, this is a **`mount`** command option at the client side, and can be changed with:

```
-o restrans=x timeo=y
```

where x is the number of retries and y the time out period

This is not necessarily a performance tuning parameter, but it can be helpful while trying to improve performance on faulty networks. At the same time, if the block sizes are increased too much, even in a good network, these options may require some attention to avoid excessive packet retransmissions.

Tip: The **`nfsstat`** command output can help you detect if your server is suffering from excessive retransmissions.

Number of NFSd instances

Most UNIX and Linux distributions start scripts for NFS services use eight instances of the nfsd daemon. The value of eight instances is used for historical reasons, as Sun decided on this number as a rule of thumb in the early days of NFS.

There is no definite answer to how many threads should start, but servers with heavier workloads tend to require more threads. A more up-to-date rule of thumb is four or eight threads per processor in the system, with one thread per CPU being the bare minimum.

However, the best approach seems to make sure that the top three nfsd instances in terms or utilization are not close to 100%. If they are, this usually is a sign that more threads can be helpful. The easiest way to check for the thread utilization is to look at the three last values of the th line in the /proc/net/rpc/nfsd file. Fortunately, unlike Samba, these threads use a considerably smaller amount of memory; each thread usually uses 1 MB of memory when heavily loaded.

Synchronous versus asynchronous behavior

Originally, Linux systems used to default to the asynchronous NFS exports behavior. In asynchronous mode, the server replies to client requests as soon as it has processed the request and handed it off to the file system. It will not wait for data to be written to storage. It provides better performance characteristics, but data corruption can occur if the server has a problem while the data is still in cache.

In the now default synchronous mode, the replies wait until the file system has finished writing the data into the storage system when using NFSv2, or it informs the client that the data is in cache and should not be discarded until further notification if using NFSv3.

The behavior is defined when the file system is exported by the **`exportfs`** command using the -o sync or -o async options. Additionally, a client is able to force a total synchronous behavior mode through the **`mount`** option O_SYNC and force a NFSv3 to behave just like a NFSv2 server in synchronous mode, but it has no effect in exported file systems that are using asynchronous mode.

For best performance, the async option is recommended, but a system administrator must understand the consequences. For extremely important and sensitive file systems, exporting them in sync mode can be a very good idea.

Keep in mind that with the current version, if the asynchronous behavior option is not included, the default behavior will be synchronous.

Conclusions

The NFS services offer a large set of parameters that can affect server performance. But, unlike Samba, some of these parameters are set up by the client at mount time.

From the server side, a couple of parameters may require attention, as the values that will yield best performance are not set by default.

Setting the server behavior to asynchronous will yield the best results. At the same time, careful monitoring of the number of nfsd instances will allow finer tuning while the server is in production.

From the client side, proper selection of the block size parameter as well as timeout and retransmission times can provide considerable performance gains.

For additional details on NFS and performance related issues, the following links can be very helpful:

http://nfs.sourceforge.net/nfs-howto/ar01s05.html
http://star-www.rl.ac.uk/star/dvi/ssn26.htx/node16.html

8.4.10 Consolidated tuning scripts and commands

Several modifications were done at the operational system level in order to improve the performance of file serving for both SUSE Linux Enterprise Server and Red Hat Enterprise Linux installs. These modifications allowed a considerable performance gain with the tested workload.

This section intends to consolidate all the commands and scripts used to tune these parameters in real time at the respective servers.

SUSE Linux Enterprise Server 10

- I/O Scheduler
  - The noop scheduler was selected. In order to choose it as the default scheduler at boot time, add the following line to the /etc/yaboot.conf file:

    `append="elevator=noop"`[4]
  - The noop scheduler is extremely simple and efficient. It does not provide any tunable parameter

[4] If the yaboot.conf file already has an append line, just add the elevetor=noop statement to the line

- File System Tuning
  - At file system creation, the log buffers were modified and moving the journaling into a different hard disk is highly recommended. This study used the journaling in the same device due to the environment used. The file system was created with the following command:

    ```
    mkfs -t xfs -l logdev=/dev/sdx,size=64m /dev/sdy
    ```

  - At file system mount time, the number of memory log buffers was changed and the access time logging was disabled. The **mount** command used to mount the file system created above is:

    ```
    mount -o noatime,logbufs=4 /dev/sdx /mounttargetdirectory
    ```

- Network Interface
  - Several network parameters were modified in order to improve the network performance. The complete script with the network related **sysctl** commands can be seen in Example 8-4.

Example 8-4 Network tuning parameters for SUSE Linux Enterprise Server

```
#Script to optimize Network parameters for test
#SUSE

echo Tuning parameters....

#Network
#-------
echo Network tuning

echo Unwanted services
#Ignore redirects
sysctl -w net.ipv4.conf.eth0.secure_redirects=1
sysctl -w net.ipv4.conf.lo.secure_redirects=1
sysctl -w net.ipv4.conf.default.secure_redirects=1
sysctl -w net.ipv4.conf.all.secure_redirects=1

#ICMP redirect
sysctl -w net.ipv4.conf.eth0.accept_redirects=0
sysctl -w net.ipv4.conf.lo.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.all.accept_redirects=0

#Disable send redirect
sysctl -w net.ipv4.conf.eth0.send_redirects=0
sysctl -w net.ipv4.conf.lo.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0
```

```
sysctl -w net.ipv4.conf.all.send_redirects=0

#Ignore broadcast pings
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
#Ignore all icmp and pings (not used in test)
# sysctl -w net.ipv4.icmp_echo_ignore_all=1
#Ignore routers invalid responses
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
echo Unwanted services: Done

echo TCP Buffers
# rmem default = "4096 87380 174760"
sysctl -w net.ipv4.tcp_rmem="32768 436600 873200"
# wmem default = "4096 16384 131072"
sysctl -w net.ipv4.tcp_wmem="8096 436600 873200"
# mem default = "196608 262144 393216"
sysctl -w net.ipv4.tcp_mem="13145728 14194304 16291456"
echo TCP Buffers: Done

echo TCP Options and connection management
sysctl -w net.ipv4.tcp_window_scaling=0
sysctl -w net.ipv4.tcp_sack=0
sysctl -w net.ipv4.tcp_dsack=0
sysctl -w net.ipv4.tcp_fack=0
#tcp_max_syn_backlog default 1024 (if more than 128MB)
sysctl -w net.ipv4.tcp_max_syn_backlog=2048
#tcp_max_synack_retries default 5 (180 s)
sysctl -w net.ipv4.tcp_synack_retries=2
#tcp_retries2 default 15
sysctl -w net.ipv4.tcp_retries2=10
#tcp_keepalive_time default 7200
sysctl -w net.ipv4.tcp_keepalive_time=3600
#tcp_max_keepalive_intvl default 75
sysctl -w net.ipv4.tcp_keepalive_intvl=50
#tcp_max_keepalive_probes default 9
sysctl -w net.ipv4.tcp_keepalive_probes=5
echo Options and connection management: Done
echo Networking tuning: DOne
```

- Memory
  - The allocated memory for the server was carefully monitored to guarantee that no paging occurred during the test, and enough memory was available for the kernel to buffer disk reads.

- During the tests, the partitions were configured with 4 GB, but measurements indicated that 2.5 GB would have been enough for the workloads tested.

- CPU
  - The amount of CPU allocated to the partition guaranteed that it was not a bottleneck. With this paradigm, SMT should stay enabled to improve performance.
  - During the CPU stress test, for the workload tests, our results indicated that 1.50 CPUs would provide the best performance/resource allocation relationship.
- Unwanted services
  - To provide additional memory and CPU cycles for the file serving daemons, the following daemons were disabled:

```
postfix
powersaved
splash_early
splash
fbset
xdm
xfs
```

Red Hat Enterprise Linux AS 4

- I/O Scheduler
  - The CFQ scheduler was selected. It is the default scheduler.
  - The CFQ scheduler provides two tunable parameters. When considering the test workload, the results indicated that the parameters should be set as follows:

```
echo 16 >> /sys/block/sda/queue/iosched/queued
echo 4 >> /sys/block/sda/queue/iosched/quantum
```

- File system
  - The file system was tuned using the **tune2fs** command to change the journaling policy. The change can be made using the following command:

```
tune2fs -o journal_data_writeback /filesystemname
```

  - At file system mount time, the number of memory log buffers was changed and the access time logging disabled. The **mount** command used to mount the file system created above is:

```
mount -o noatime /dev/VolGroup00/fsname /mounttargetdirectory
```

- Network interface
  - Several network parameters were modified in order to improve the network performance. The complete script with the network related **sysctl** commands can be seen in Example 8-5.

Example 8-5 Network tuning parameters for Red Hat Enterprise Linux

```
#Script to optimize Network parameters for test
# RedHat
echo
echo Network tuning
echo Network unwanted services:

#Unwanted services
#Ignore redirects
sysctl -w net.ipv4.conf.eth0.secure_redirects=1
sysctl -w net.ipv4.conf.lo.secure_redirects=1
sysctl -w net.ipv4.conf.default.secure_redirects=1
sysctl -w net.ipv4.conf.all.secure_redirects=1

#ICMP redirect
sysctl -w net.ipv4.conf.eth0.accept_redirects=0
sysctl -w net.ipv4.conf.lo.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0
sysctl -w net.ipv4.conf.all.accept_redirects=0

#Disable send redirect
sysctl -w net.ipv4.conf.eth0.send_redirects=0
sysctl -w net.ipv4.conf.lo.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0
sysctl -w net.ipv4.conf.all.send_redirects=0

#Ignore broadcast pings
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
#Ignore all icmp and pings (not used in test)
# sysctl -w net.ipv4.icmp_echo_ignore_all=1
#Ignore routers invalid responses
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
echo Network unwanted services: Done

echo TCP Buffers:
#TCP Buffers
# rmem default = "4096 87380 174760"
sysctl -w net.ipv4.tcp_rmem="32768 436600 873200"
# wmem default = "4096 16384 131072"
```

```
sysctl -w net.ipv4.tcp_wmem="8096 436600 873200"
# mem default = "3145728 4194304 6291456"
sysctl -w net.ipv4.tcp_mem="13145728 14194304 16291456"
echo TCP Buffers: Done

echo TCP Options and connection management
#TCP Options and connection management
sysctl -w net.ipv4.tcp_window_scaling=0
sysctl -w net.ipv4.tcp_sack=0
sysctl -w net.ipv4.tcp_dsack=0
sysctl -w net.ipv4.tcp_fack=0
#tcp_max_syn_backlog default 1024 (if more than 128MB)
sysctl -w net.ipv4.tcp_max_syn_backlog=2048
#tcp_max_synack_retries default 5 (180 s)
sysctl -w net.ipv4.tcp_synack_retries=2
#tcp_max_retries2 default 15
sysctl -w net.ipv4.tcp_retries2=10
#tcp_max_keepalive_time default 7200
sysctl -w net.ipv4.tcp_keepalive_time=3600
#tcp_max_keepalive_intvl default 75
sysctl -w net.ipv4.tcp_keepalive_intvl=50
#tcp_max_keepalive_probes default 9
sysctl -w net.ipv4.tcp_keepalive_probes=5
echo TCP Options and connection management: Done
echo Network Tuning: Done
```

- Memory
  - The allocated memory for the server was carefully monitored to guarantee that no paging occurred during the test, and enough memory was available for the kernel to buffer disk reads.
  - During the tests, the partitions were configured with 4 GB, but measurements indicated that 2.5 GB would have been enough for the workloads tested.
- CPU
  - The amount of CPU allocated to the partition guaranteed that it was not a bottleneck. With this paradigm, SMT should stay enables to improve performance.
  - During the CPU stress test, for the workload tests, results indicated that 1.50 CPUs would provide the best performance/resource allocation relationship.

- Unwanted services
  - To provide additional memory and CPU cycles for the file serving daemons, the following daemons were disabled:
    - autofs
    - sendmail
    - isdn
    - pcmcia
    - xdm
    - xfs

8.5 Print server workload

Print servers remove the requirement to install printers on individual clients and are capable of supporting a large number of printer types and print queues. They manage client print requests by spooling the print job to disk.

The printer device itself will be the major bottleneck that will influence performance due to the slow printing speeds when combined with limited memory capacity that will make a service request take longer to produce an output while utilizing resources on the print server. Therefore, the critical areas that impact performance are the speed of the printer, that in general are orders of magnitude below the System p capacity, to transfer data from memory or disk to the device.

By default, the spool directory is located on the same disk as the operating system files. To avoid a performance impact on the system as a whole, this should be located on a physical drive other than the operating system disk.

Implementing printer pools and virtual printer configurations may help to reduce printing workload.

In addition, the printer connection is a possible bottleneck. Printers that connect to a parallel or serial port requires much CPU time, because these ports are very inefficient. If possible, the usage of a network attached printer can reduce the CPU cycles used by the printing system considerably.

Finally, a good practice is to create multiple logical printers to the same device so you can schedule different work times. For example, you can direct a huge document to a logical printer that works from 12:00 AM to 6:00 AM and let the smaller documents be more efficiently printed throughout business hours. If you

organize the time based on the document size and priority, you can significantly reduce the print traffic, and therefore the CPU cycles needed to dispatch it.

9

Apache Web server

This chapter discusses the Web serving workload and the performance analysis. We will mainly focus on the Open Source Apache HTTP server.

9.1 General information about Apache

A Web server is a very common application, and on the Linux platform, Apache is undoubtedly the most popular Web server software.

The current stable Apache release series is 2.2.x (at the time this book was written). The last stable series is V2.0, and the earlier one is V1.3. Red Hat Enterprise Linux AS 4 comes with Apache 2.0.x, SUSE Linux Enterprise Server 9 comes with V1.3.x and V2.0.x, and SUSE Linux Enterprise Server 10 comes with V2.2.x.

Apache 2.0 was a major rewrite of Version 1.3, and it provides better performance, scalability, and supports more platforms. Apache 2.2 has many improvements compared to V2.0, but not as many improvements as with V2.0 compared to V1.3. In this chapter, when we mention Apache, we are mostly referring to Apache 2.2 and 2.0.

9.1.1 Apache architecture models

There are two architecture models supported by Apache 2.0:

- Process-driven architecture model

  The process-driven (or *fork*) architecture creates a separate process to handle each connection. Each new process is a copy of the original process. When started, Apache creates several new child processes to handle Web server requests in addition to always keeping a set number of processes idle to handle peak demand, as shown in Figure 9-1.

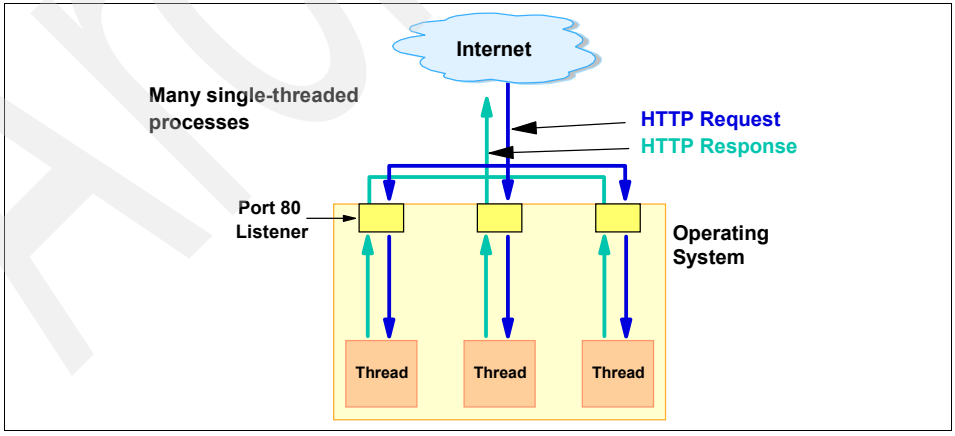


Figure 9-1 Web server based on a process-driven model

- Multi-threaded architecture model

  Apache V2.0 offers the option of using a second model, the multi-threaded architecture. According to the Apache Foundation, it should improve scalability for many configurations.

  With this model, only two processes are created to handle all requests. Within one of the processes, threads are created to handle server requests.

  A thread (also called a lightweight process) is a stream of control that can execute its instructions independently. More simply put, a thread is a unit of execution sharing resources with other threads within a process, as shown in Figure 9-2.

  The multi-threaded model is theoretically a much more efficient architecture implementation and you will see performance improvements with most Apache workloads.

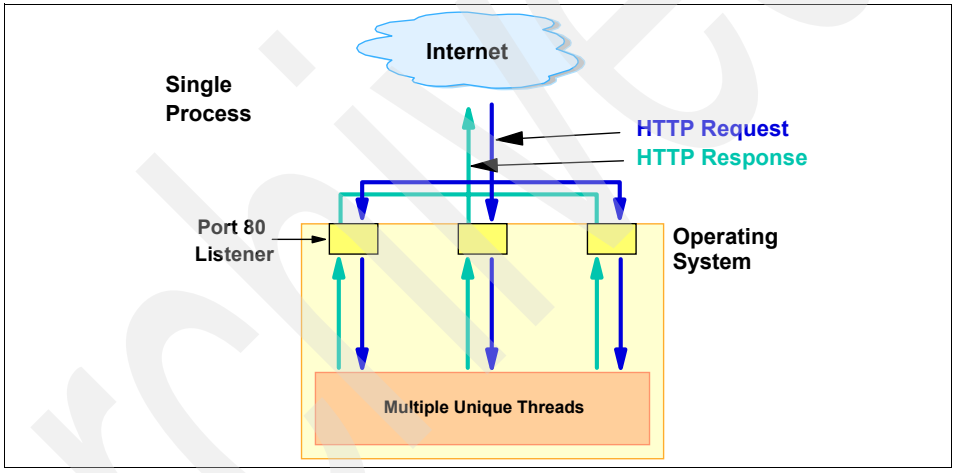


Figure 9-2 Web server based on a multi-threaded model

9.2 Potential bottlenecks

Web sites have become more and more complex today and the content served to the client is quite different; some are just static pages and images, some are media files, and some are dynamically generated according to user input. They will have different needs for the CPU, memory, disk, and network subsystems.

The performance of a Web server is usually measured by the following parameters:

- Throughput

  Throughput is how fast the Web server can serve the clients. It can be measured using two different units. In both cases, the larger the number, the better.

  - Requests per second

    This is usually the first number you should review when benchmarking a Web server.

  - Bytes transmitted per second

    This information is the bandwidth of the Web server, and will tell you if the Web server is saturating the wire.

- Latency

  This is also known as the response time. This is the time that elapses between the client request being made and when the results starting to come in from the Web server. The latency is usually the summation of the time that the request comes in and response spent traveling through the network, and the time the Web server is used to process the request and generate the result. So if the network is in good condition, an increase in the latency will tell you that the Web server just spent more time to process the request. A smaller value would be better.

- Concurrent requests

  This is how many client requests the Web server can handle simultaneously. This is an important number, because a Web site is usually viewed by many browsers and a server with a higher concurrent request capability can support more clients, so the server hardware is utilized more efficiently.

So the result we expected after tuning the Web server is to support more concurrent requests, or larger throughput, or lower latency.

9.2.1 Important subsystems

The way the Web server is affected by each subsystem depends on the type of content it serves:

- Mainly static content

  Since serving static content is just a matter of sending data over the network, the network subsystem would be the most important component. The subsystem importance list is as follows:

  - Network
  - Memory
  - CPU

- Dynamic content

  To generate dynamic content, the Web server process usually needs to run some programs or scripts, and sometimes fetch data from a database, process it, and then build the pages. The programs or scripts will consume memory and CPU. The subsystem importance list is as follows:

  - Memory
  - CPU
  - Disk
  - Network

- Secure content

  Security is important for today's business Web sites. To protect the information, the data is encrypted into secured messages before they go into the untrusted network, and the receiver just decrypts the secured messages and gets the original data. The encrypt process is CPU sensitive, and under heavy load, it will largely decrease the throughput. So the subsystem importance list is:

  - CPU
  - Memory
  - Disk
  - Network

A real world Web site is a mix of the three aforementioned typical content types, so it not a simple process to define the most important subsystem. As a rule of thumb, Apache will run out of memory before anything else when servicing mixed content. The amount of memory significantly influences the performance of the Web server. The more memory you have, the more the server can cache the data requested and serve it to users faster than if the data had been on disk.

Generally speaking, the Web server should never have to page to disk on a regular basis. We strongly recommend that you monitor the usage of the swap file closely to be alerted of frequent paging.

9.2.2 Monitoring Apache status

Please refer to Chapter 5, “Linux monitoring tools” on page 87 for common Linux monitoring tools.

Here are some Apache specific monitoring tools.

Apache mod_status module

This mod is available as a standard Apache module included in the software package.

The mod_status module provides information about server activity and performance; an HTML page is presented that gives the current server statistics in an easily readable form.

The details given are:

- The number of workers serving the requests.
- The number of idle workers.
- The status of each worker, the number of requests that the worker has performed and the total number of bytes served by the worker.
- The total number of accesses and byte count served.
- The time the server was started/restarted and the time it has been running for is the average of the number of requests per second, the number of bytes served per second, and the average number of bytes per request.
- The current percentage CPU used by each worker and in total by Apache.
- The current hosts and requests being processed.

More details about how to configure and use this module are available in the Apache manual.

Attention: Since this module provides server status and client connection information, you should carefully limit access to only trusted viewers.

Apachetop

The tool is available at:

http://www.webta.org/projects/apachetop/

Apachetop is a top-like display for Apache information, including requests per second, bytes per second, most popular Web addresses, and so on. Apachetop watches a log file generated by Apache and generates human readable output in real time.

Figure 9-3 shows a sample output of Apachetop.

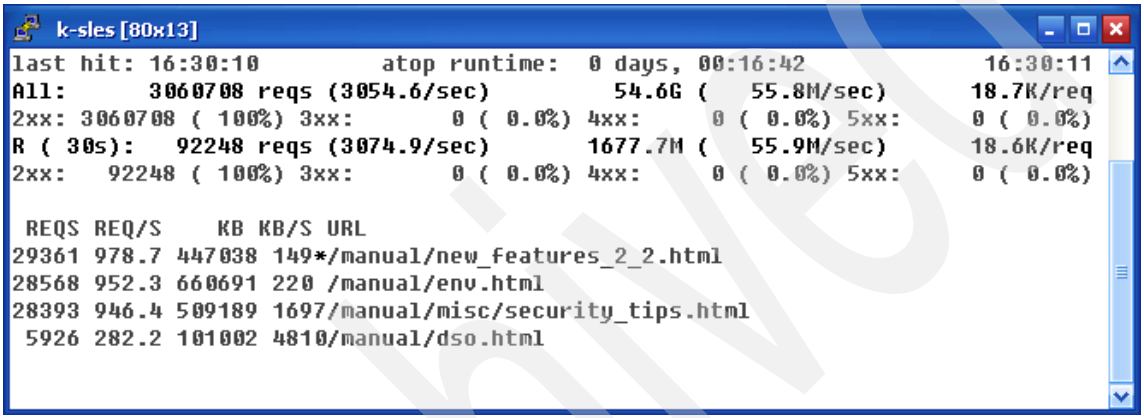


Figure 9-3 Screen of apachetop command

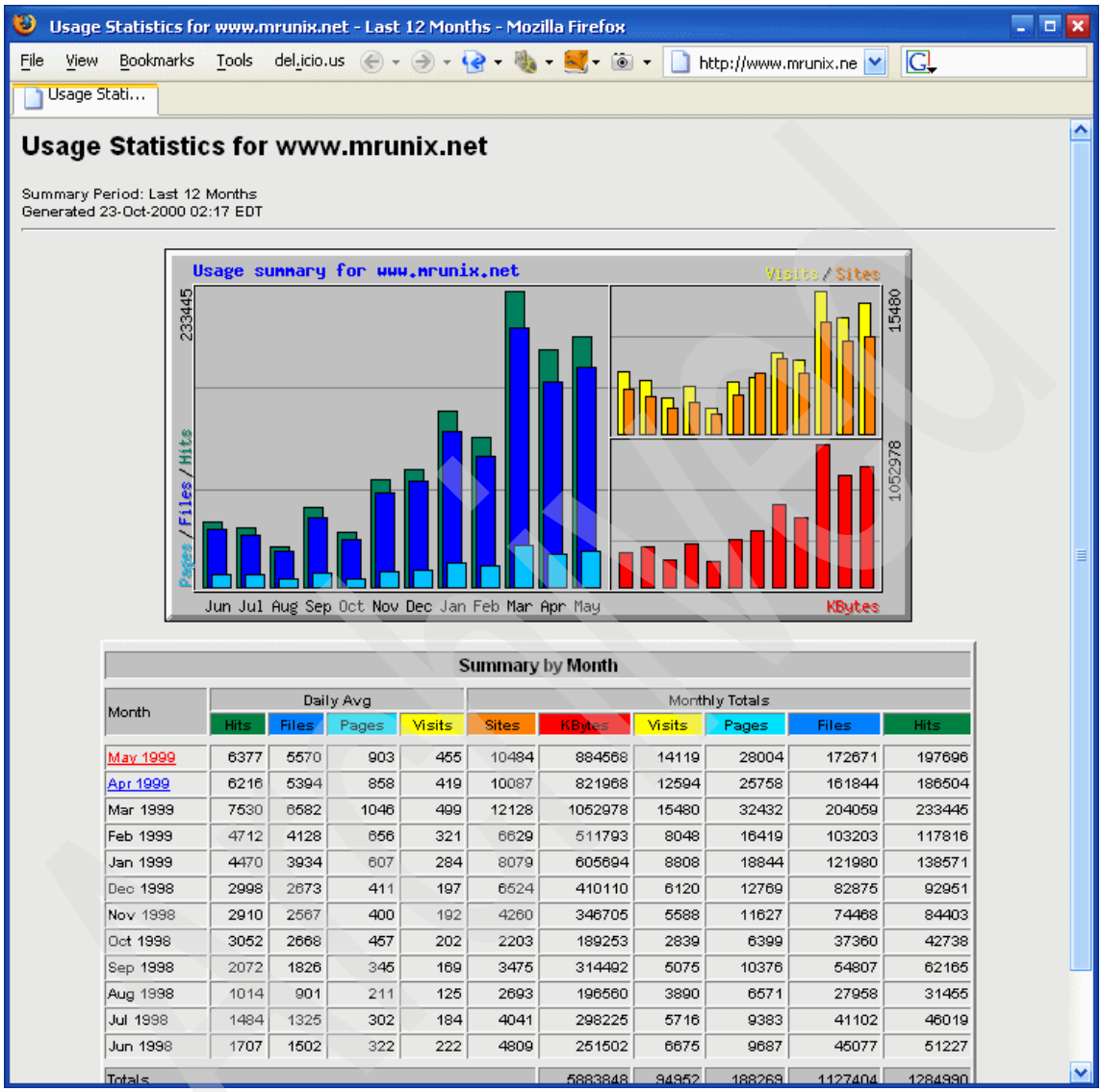
Webalizer

This tool is available at:

http://www.mrunix.net/webalizer/

Webalizer is a Web server log file analysis tool. It produces highly detailed, easily configurable usage reports in HTML format, for viewing within a standard Web browser. It is not a real time state viewer like Apachetop; instead, it is good for monitoring and analyzing long run statistic data.

Figure 9-4 shows a report produced by Webalizer.



Summary by Month

Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	KBytes	Visits	Pages	Files	Hits
May 1999	6377	5570	903	455	10484	884568	14119	28004	172671	197696
Apr 1999	6216	5394	858	419	10087	821968	12594	25758	161844	186504
Mar 1999	7530	6582	1046	499	12128	1052978	15480	32432	204059	233445
Feb 1999	4712	4128	656	321	6629	511793	8048	16419	103203	117816
Jan 1999	4470	3934	607	284	8079	605694	8808	18844	121980	138571
Dec 1998	2998	2673	411	197	6524	410110	6120	12769	82875	92951
Nov 1998	2910	2567	400	192	4260	346705	5588	11627	74468	84403
Oct 1998	3052	2668	457	202	2203	189253	2839	6399	37360	42738
Sep 1998	2072	1826	345	169	3475	314492	5075	10376	54807	62165
Aug 1998	1014	901	211	125	2693	196560	3890	6571	27958	31455
Jul 1998	1484	1325	302	184	4041	298225	5716	9383	41102	46019
Jun 1998	1707	1502	322	222	4809	251502	6675	9687	45077	51227
Totals						5883848	94952	188269	1127404	1284990

Figure 9-4 A report produced by webalizer

AWStat

This tool is available at:

http://awstats.sourceforge.net/

AWStats is a free, powerful, and full featured tool that generates advanced Web, streaming, FTP, or mail server statistics graphically. Like Webalizer, it is a long-term monitor.

Figure 9-5 shows a report produced by AWstats.

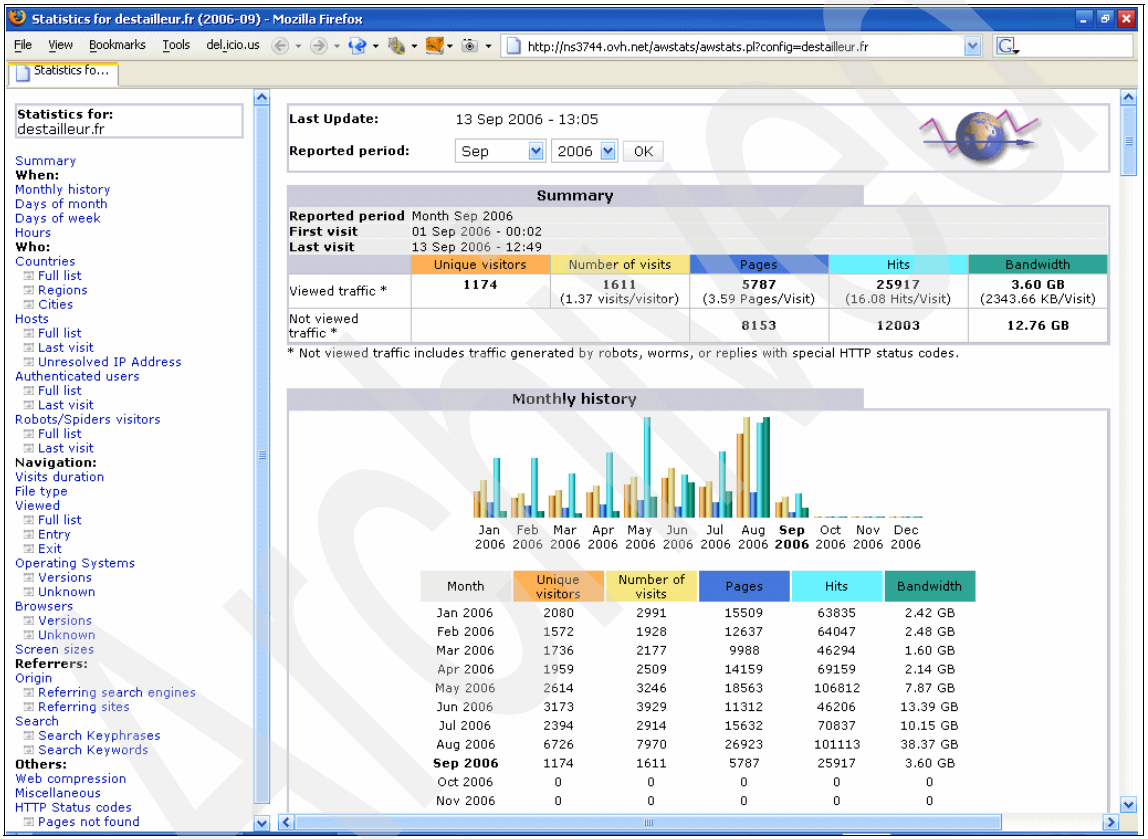


Figure 9-5 A report produced by AWstats

SNMP Apache Module

This tool is available from:

http://mod-apache-snmp.sourceforge.net/english/index.htm

The SNMP Apache Module allows you to monitor different configuration and status values of the Apache Web server using Simple Network Management Protocol (SNMP).

Some Apache values that can be monitored using this module:

- Total Traffic
- Total Access
- Last Restart
- Uptime
- HTTP errors (400, 403, 404 and so on)
- Status (On, Off, Not response)

9.3 Tuning Apache

Red Hat Enterprise Linux 4 and SUSE Linux Enterprise Server 10 both have Apache included in the distribution and have reasonably good options set in the configuration files. However, the default configuration may not suitable for every type of the workloads we described in 9.2.1, “Important subsystems” on page 269. Sites that mostly serve dynamic content obviously should put more tuning efforts into CPU and memory, while sites serving static content should care about network and disk. In this section, we will discuss the Linux system and Apache tuning options by each major subsystem, so you could easily apply them to your site according to the bottlenecks that you may encounter.

9.3.1 Scenario briefing

The scenario we use here is a SUSE Linux Enterprise Server 10 system running on a partition on a OpenPower 720. The partition has a 1-core processor, 4 GB memory, 20 GB virtual disk, and a virtual NIC.

There are several charts in this section that visually show the performance or system resources consumed, before and after we apply the tuning options. The data that is used to generate these charts are measured in the testing environment we introduced above; it does not imply that the result will be the same in your system.

9.3.2 General optimization

Here we discuss general optimization options.

System ulimit

The /etc/security/limits.conf file lets you specify a variety of limits:

- How many processes and child processes a user can open?
- How much memory can a user consume using soft and hard limits?
- Maximum number of open files?
- Maximum CPU time?
- Maximum size locked in memory address space?
- Maximum stack size?

To ensure Apache processes have enough resources to grow, add these lines to the /etc/security/limits.conf file:

```
apache  soft  nproc   2048
apache  hard  nproc   4096
apache  soft  nofile  262144
apache  hard  nofile  262144
```

Bash shell users can use following command to check the current ulimit value:

```
ulimit -a
```

C shell users can just use the **limit** command.

Disable unused modules

Apache is a highly modularized software, and only the processing core is built statically into the main executive; most functions can be built as individual modules, as separated files on the file system, and are loaded on demand. For each module loaded by Apache, it will provide certain functions, and of course, consume system resources, most notably system memory. So we can remove some of the modules if the functions they provided are not necessary.

Removing a module is as simple as just commenting out the specific LoadModule directive in the Apache configuration file. For example, if you do not need CGI support, then just comment the line as shown below:

```
LoadModule cgi_module modules/mod_cgi.so
```

Some modules may provide useful functions, but also requires extra performance time; an example is mod_status with the ExtendedStatus directive set to On. The time required may be minimal, but if you need the highest performance, set it to Off or disable mod_status entirely.

Note: Modules may have interdependency. Disabling a module that is not useful may accidently break another module. Make sure you test the configuration before using it in production.

9.3.3 Optimize processor, memory, and process execution

To serve many client requests, Apache can take advantage of multi processor and large memory systems efficiently. It uses the multi-process or multi-thread model described in 9.1.1, “Apache architecture models” on page 266. You must properly choose and configure the Multi-Processing Module (MPM) to improve the performance.

Select the proper MPM

The two different MPMs, process based *prefork* and thread based *worker*, have different characteristics. Worker generally is a good choice for high-traffic servers, because it has a smaller memory footprint than the prefork MPM. On many systems, prefork is comparable in speed to worker, but it uses more memory.

Selecting MPM for Apache is an easy task. If you compile Apache from the source code, just use the argument --with-mpm=*NAME* when invoking the configuration script. If you use the package that comes with Red Hat Enterprise Linux 4 or SUSE Linux Enterprise Server 9 and 10, then they have the two MPMs included.

For Red Hat, the two MPMs are included in the main Apache package, named httpd. To choose a MPM, edit the file /etc/sysconfig/httpd and find the line:

```
#HTTPD=/usr/sbin/httpd.worker
```

To use worker MPM, just un-comment it. Leave it commented to use prefork.

For SUSE Linux Enterprise Server, the two MPMs are in two separate packages, apache2-prefork and apache2-worker. Install one of the two packages to select the corresponding MPM. If both are installed, edit the file /etc/sysconfig/apache2 and find the line:

```
APACHE_MPM=""
```

Modify it to install MPM.

Figure 9-6 shows the difference between the prefork and worker MPMs in terms of CPU and memory consumption. The worker MPM clearly consumes less CPU and memory than prefork.

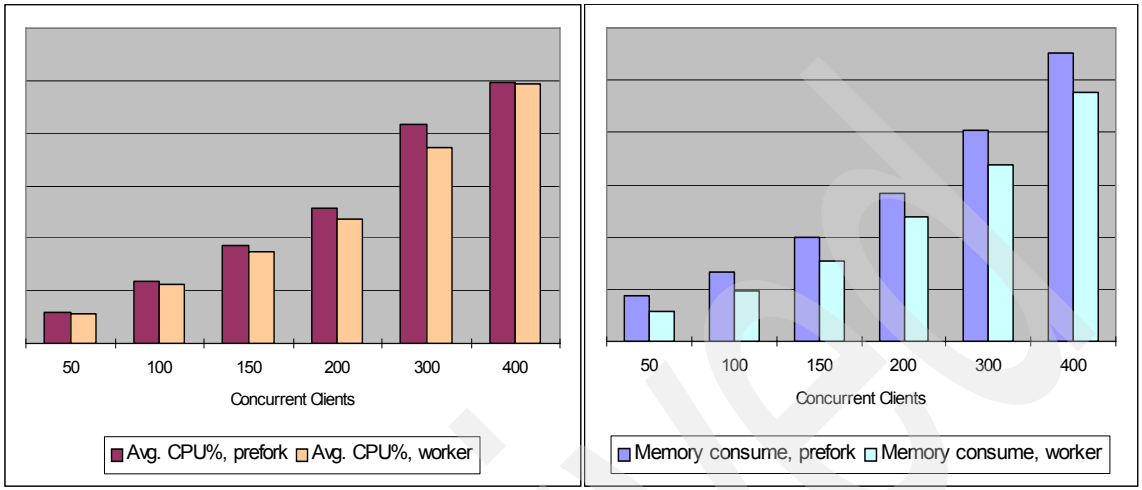


Figure 9-6 CPU and memory resource consumption comparisons between "prefork" and "worker" MPMs

Worker MPM has a drawback that it is not thread safe for some Apache modules, so they will not run worker. One familiar example is PHP. In the official PHP document, the suggestion is that mod_php should be used with prefork MPM.[1] However, you still can use PHP script with worker MPM; just set up PHP support as FastCGI instead of the Apache module.

Configure MPM

The following directives relate to the Multi-Processing Module (MPM) and affect how many and how fast Apache can process client requests.

- StartServer

  This is the number of child processes Apache will create on startup. This setting only applies at startup. After startup, the number of child processes is dynamically controlled and is dependent on other settings. If you have to restart your server frequently, or while servicing requests, it is a good idea to increase this number so the server will get back up to speed quickly.

[1] http://www.php.net/manual/en/faq.installation.php#faq.installation.apache2

The default value is 5, but should be set close to the average number of child processes while under normal load to minimize startup delay times. The algorithm to create new processes in Apache uses a minimum delay of one second before creating a new process, so the number of processes created is doubled every second until it reaches 32 processes per second or until the load can be handled without having to create new processes.

- MinSpareServers/MinSpareThreads

  This setting specifies the minimum number of idle child processes that must be available to service new connections at any given time. The pool of available processes is updated by Apache to remain at least equal to that limit when the connection number increases. These processes are useful when you are experiencing spikes.

  The default value for this parameter is 5. For heavily used sites experiencing a lot of spikes, it should be increased to 25. This will reduce the latency time users are experiencing when connecting during a climbing load period.

- MaxSpareServers/MaxSpareThreads

  This setting defines the maximum number of idle child processes that can be made available to service new connections at any given time. This pool of processes is updated by Apache to remain between the minimum and the maximum values when the connection number increases or decreases. Having too many idle processes, however, is a waste of resources. If the number of processes available in the idle pool exceeds this value, Apache will terminate the excess processes.

  The default value for this parameter is 20, but for a busy server with enough memory available, it could be higher; 100-150 is reasonable.

- MaxClients

  This parameter defines the maximum number of child processes available simultaneously. That is, it is the maximum number of requests that will be served at one time.

  The maximum value that can be set is 256. If you need a value higher than that, you must also raise the value of *ServerLimit*.

  For servers mostly serving static content, you can set it to several hundreds or even one thousand, based on your processor core number and memory capacity. A reasonable start may be 200-400, and you can raise it by steps of 50-100 if the system has spare CPU and memory resources.

If your site includes many dynamic pages and you do increase this value, you may start to experience memory problems and swapping. In these circumstances, you should consider reducing the value. The incoming requests, instead of being processed immediately, will be put on a queue until the process becomes available. This will work faster than having to swap memory to disk. However, a better solution would be to increase the amount of RAM installed.

- ServerLimit/ThreadLimit

  These directive set the maximum configurable value of MaxClients (default of 256). MaxClients normally cannot be set to a larger value unless you raise this one accordingly. Take care when you use this directive, because if it is set to a value much higher than necessary, extra shared memory will be allocated and it may go unused. If both ServerLimit and MaxClients are set to values higher than the system can handle, that is, out of the system resource capacity, Apache may not start or become unstable. The value has a hard limit of 20000.

- MaxRequestsPerChild

  This directive controls how many requests a process will serve before exiting. The default, 0, causes the process never to exit. However, accepting this value of 0 could lead the system to produce memory leaks if you use poorly written modules. So only set it to 0 if you have your Web server well tested and remember to keep monitoring the memory utilization to determine if memory leaks are occurring. For safety, set it to a large value like 10000.

Dynamic content

Serving dynamic content is usually accomplished by running scripts on the Web server, the scripts process the request from the client, which sometimes need to fetch data from a database, and then send the responses back to client. The execution of the scripts will consume memory and CPU resources, especially in a high traffic Web site, and the dynamic content will place a heavy load on the processor and memory subsystem.

There are several way to optimize the situation. Since the scripts themselves are programs, the programmers can optimize the code. And we also can tune the Web server to make the execution more effective.

- Migrate Common Gateway Interface (CGI) program to script module

  The classic way of serving dynamic content is running by a CGI program. A CGI program can be any executable on the host system written by any language, but it is usually written by scripting languages like Perl. It is easy to support CGI in Apache and there are many CGI applications around the internet. However, CGI has a very resource consumptive design: for every single request to the CGI program, the Web server has to spawn a new

process. It is a very expensive operation, wastes a great deal of time waiting for the process to be ready, and the system can easily have hundreds of CGI processes running if the site is busy.

Fortunately, many scripting languages that are commonly used as CGI engines have an Apache module that can run the scripts just within an existing Apache process, so there is no need to spawn a new one. Table 9-1 shows the scripting languages and their Apache modules.

Table 9-1 Scripting language and their Apache modules

Scripting language	CGI interpreter	Apache module
Perl	/usr/bin/perl	mod_perl[a]
PHP	/usr/bin/php	mod_php[b]
Python	/usr/bin/python	mod_python[c]
Ruby	/usr/bin/ruby	mod_ruby[d]

a. Comes with Red Hat Enterprise Linux 4 as mod_perl and SUSE Linux Enterprise Server 10 as apache2-mod_perl.
b. Comes with Red Hat Enterprise Linux 4 in the php package and SUSE Linux Enterprise Server 10 as apache2-mod_php5.
c. Comes with Red Hat Enterprise Linux 4 as mod_python and SUSE Linux Enterprise Server 10 as apache2-mod_python.
d. Neither Red Hat Enterprise Linux nor SUSE Linux Enterprise Server include it. Available from http://modruby.net/

So whenever possible, you should use the Apache module to run those scripts instead of CGI. Most modern Web applications have abandoned the CGI way and support to run the Apache module.

- Migrate CGI to FastCGI

  While running scripts using the Apache module is good, there are still some old programs that run very well using CGI and have not migrated to support using the Apache module. For these kind of programs, there are several enhancement efforts to the CGI standard, such as FastCGI and Speedy CGI.

  The FastCGI and Speedy CGI way is to have a persistent process running to handle the incoming requests; in contrast, the plain CGI needs a new process for each request. FastCGI even allows the CGI programs to run on a different host, separate from the Web server, so that a busy site can have the load distributed to more hardware.

- Use script accelerator

  When the scripts are invoked, either in CGI mode or by the embedded module in Apache, there are several process steps:

  a. Load the script file from disk.

  b. The scripting engine compiles the content of the file, and translates it into the code that can execute directly.

  c. Let the translated code execute in the scripting engine and process the input from clients.

  d. Send the result back to the Web server, which lets it go through its internal pipe to do more processing.

  e. The Web server sends the final response back to the client.

  Usually the second step is the slowest, and is a waste, since the content of the script file does not change most of the time, especially in a production environment. So if we can have the result of second step cached, the time needed to read and compile the script is saved. PHP is the language that already have several of these kind of accelerators implemented:

  – Zend Optimizer, from Zend Technologies Ltd, available at:

    `http://www.zend.com/products/zend_optimizer`

  – Turck MMCache, available at:

    `http://turck-mmcache.sourceforge.net/`

    Its development is stalled and eAccelerator is the successor.

  – eAccelerator, available at:

    `http://www.eaccelerator.net/`

  – Alternative PHP Cache, available at:

    `http://pecl.php.net/package/APC`

  All these accelerators can provide very good results for PHP script, although the result may be different for different applications, but usually can be several times faster.

Figure 9-7 shows the performance gain of PHP scripts running with eAccelerator. The accelerated result is normalized against the non-accelerated data, we can see that the average page response time dropped dramatically, and the network throughput increased four times, which means four times more dynamic Web pages are served.

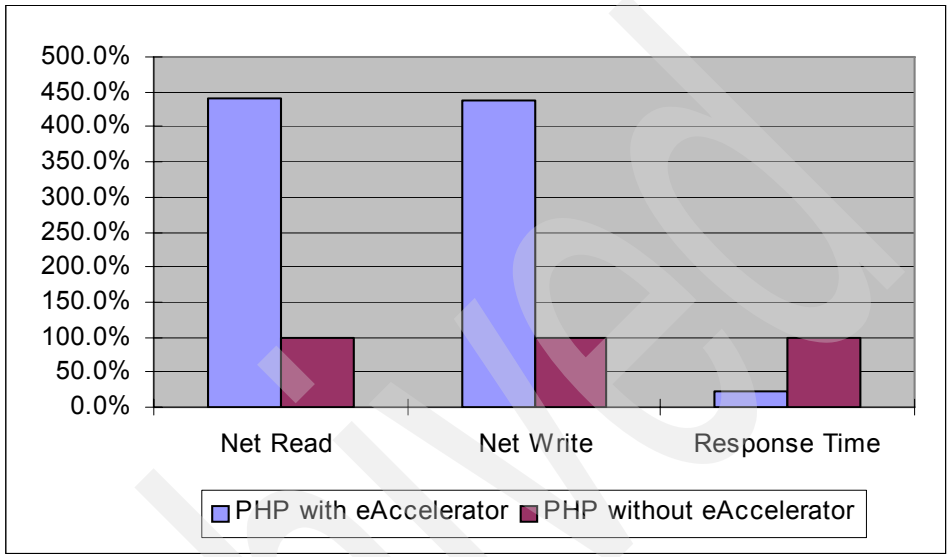


Figure 9-7 Performance gain of PHP script with accelerator

9.3.4 Optimize file system and storage access

Reading files from disk is always a slow operation, so we should tune the system and Apache to reduce the read from disk as much as possible. Having enough memory is always a good idea, since the Linux system could put the file content in memory for the first read, and all subsequential read access to the same data can access the cache directly instead of executing the expensive disk access operation again. Apache comes with its own cache mechanism that can also make use of the memory to cache disk data.

Let us look at the following tuning practices that intended to optimize the Linux system and Apache to have better file system and disk access results.

Maximum number of file handles

The maximum file handles that Linux supports will impact how many pages Apache can serve simultaneously. The following command displays the current maximum:

```
# sysctl fs.file-max
fs.file-max = 131063
```

An acceptable value may be 256 KB or more. To set this value, use the following command:

```
sysctl -w fs.file-max=262144
```

You need to put the directive in /etc/sysctl.conf for it to become effective after system reboot.

Reduce disk access

- File access time stamp

  Linux records the time when a file was last modified or accessed. There is a cost associated with it, and disabling this feature can eliminate the extra cost.

  To disable this feature, put the option noatime into the /etc/fstab on the line related to the file system.

  - For the Ext3 file system:

    Before `LABEL=/ / ext3 defaults 1 1`

    After `LABEL=/ / ext3 defaults,noatime 1 1`

  - For the ReiserFS file system:

    Before `/dev/sda2 / reiserfs defaults 1 1`

    After `/dev/sda2 / reiserfs defaults,noatime 1 1`

- AllowOverride

  This directive specifies whether the .htaccess file is to be read to determine access authority. If this option is enabled, Apache will attempt to open .htaccess for each file name component. For example, for the following configuration:

  ```
  DocumentRoot /www/htdocs
  <Directory />
  AllowOverride all
  </Directory>
  ```

For every request to the URL like /index.html, Apache will try to open /.htacess, /www/.htaccess, and /www/htdocs/.htaccess to read in options. To prevent this action, use this directive:

```
AllowOverride None
```

And put all the configurations in the main Apache configuration file.

Logging

Logging is an expensive but sometime necessary operation. Each entry has to be written to the log files, and on a busy site, this could mean thousands of entries per minute, a lot of disk space, and a number of significant CPU cycles. There are, with the default configuration, two log files available with Apache:

- Access log

  This log gets a new entry each time a request is received. This is equal to about 1 MB for every 1000 requests. The access log is a good tool if you wish to track who is using your site and which files are mostly requested. An example of a log entry is:

  ```
  127.0.0.1 - - [24/May/2002:12:15:11 -0700] "GET /apache_pb.gif
  HTTP/1.1" 304 0
  ```

  If you want to maximize the performance of your server, you should turn access logging off by commenting out the CustomLog entry from your configuration file (put a # in front of it). See Example 9-1 for details.

Example 9-1 Access log location in httpd.conf

```
# The location and format of the access logfile (Common Logfile
Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here. Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
CustomLog logs/access.log common
```

- Error log

  There are eight levels of logging available for the error log, as shown in Figure 9-8. The default setting is warn, which should give you all the information you need about your server in normal operating mode.

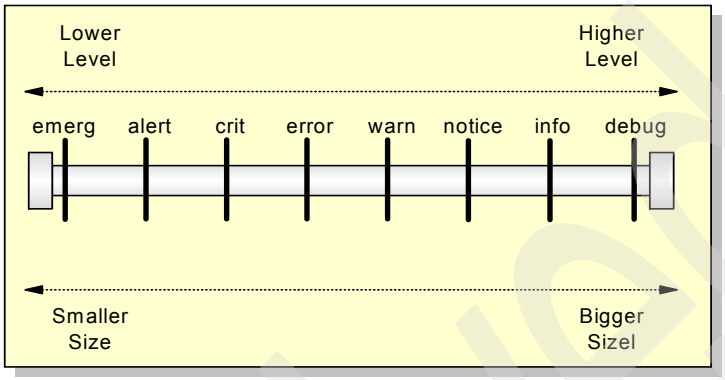


Figure 9-8 Levels of logging for the error log file

The error log is a helpful tool when you are experiencing problems with your Web server, but it can be costly under a high load. To minimize the logging, set the level to error. But do not set it to lower than crit, since you will need the log to alert you if there is any trouble.

Use the following directive to set the error log level:

```
LogLevel error
```

Setting the LogLevel to debug can help you diagnose problems. Apache will generate a great deal of useful information (see Example 9-3 on page 289). Do not forget to set it back to a higher level to reduce the logging load.

Please consult the Apache manual for details of each log level. An online version is here:

`http://httpd.apache.org/docs/2.0/mod/core.html#loglevel`

Although the logging operation will produce much disk I/O for a high traffic Web site, and the log file can occupy much disk space (for a busy site, it can easily grow to several gigabytes in a few hours), we still do not recommend that you turn it off completely, since log files will record information that is extremely useful for the following purposes:

- Monitoring the server status
- Monitoring site traffic and user requests for analysis or accounting purposes
- For security analysis

A workaround is to save the log file on a separate disk drive or separate logic disk in a array, other than the same disk that holds the Web site content. It will help to distribute the disk I/O load; however, Apache still needs to spend time for writing log files. You have to balance the logging and performance requirements for a real world case.

Use caching modules

Apache 2.0 includes a series of modules that bring caching capabilities to the Web server. Using caching modules can give you up to a 100% boost in the number of static files served within the same time interval.

The online documentation is available from:

http://httpd.apache.org/docs/2.0/mod/mod_cache.html

Memory caching for Apache 2.0 requires two modules, one main module, mod_cache, and one responsible for in-memory caching, mod_mem_cache.

mod_cache directives

The following directives apply to the main module (mod_cache):

- CacheDefaultExpire

  This is the default time in seconds that an entry will stay in cache without expiring. This default value will be used if you do not use the “Expires” or the “Last-Modified” entities in the HTTP header. The default value is 3600 seconds, which represents one hour.

  ```
  CacheDefaultExpire 43200
  ```

- CacheMaxExpire

  This is the maximum time, in seconds, that an entry will stay in cache without expiring. This default value is used to ensure that the document is out of date even when an expiry date is provided by the “Expires” entity in HTTP header. The default value is 86400 seconds, which represents 24 hours. This directive has precedence over the previous one based on the Last-Modified entity of the HTTP header.

  ```
  CacheDefaultExpire 252000
  ```

  We encourage you to use the default value for CacheMaxExpire or to set it to a large interval of time, especially if your Web site is stable and does not carry file changes often. Setting that value to a small interval will cause the client to retrieve the document even if it does not change at all, and therefore degrade the performance of your site.

- CacheEnable and CacheDisable

  These directives instruct the caching modules to allow or deny caching of URLs above the URL string pass in a parameter. In both cases, the type of caching needs to be set in the argument line (mem for memory in our case).

  ```
  CacheEnable mem /Webtree/tractors
  CacheDisable mem /Webtree/wheel_loaders
  ```

- CacheIgnoreCacheControl

  This directive will let you cache and serve from the cache files that the client is trying to always get fresh from the disk by using the no-cache or no-store parameter in the request. By default, this value is Off, but should be changed to enforce a greater number of requests served from the cache.

  ```
  CacheIgnoreCacheControl On
  ```

mod_mem_cache directives

The following directives apply to the memory caching module and will configure the limits of the cache.

- MCacheSize

  This is the maximum amount of memory used by the cache, in KB. The default value is 100 KB. You should first determine the size of the Web sites you are hosting, then the amount of memory you have on your server, and finally set this directive in accordance with these values.

  ```
  MCacheSize 60000
  ```

- MCacheMaxObjectCount

  This is the maximum number of objects to be placed in the cache. If you want to cache all the static files within your Web tree, just set that to a higher value, and then set the number of files your server is servicing. The default value is 1000 objects.

  ```
  MCacheMaxObjectCount 6500
  ```

- MCacheMinObjectSize

  This is the minimum size, in bytes, that an object has to be in order to be eligible for caching. This is used if you do not want small objects to be cached and reach the maximum number of objects without filling the memory space. Having many objects in the cache could also increase the search time for each URL in the hash array. The default value is 0 bytes.

  ```
  MCacheMinObjectSize 0
  ```

- MCacheMaxObjectSize

  This is the maximum size in bytes that an object must have to be eligible for caching. This is used if you do not want large files to be cached in the event of a small memory situation. The default value is 10000 bytes, but should be set much higher if you do not have memory limitations.

  ```
  MCacheMaxObjectSize 580000
  ```

Example 9-2 includes a set of directives to enable and maximize cache utilization.

Example 9-2 Example of a set of configuration for caching modules

```
LoadModule cache_module modules/mod_cache.so
<IfModule mod_cache.c>
 CacheOn On
 CacheMaxExpire 172800
 CacheIgnoreCacheControl On
 LoadModule mem_cache_module modules/mod_mem_cache.so
 <IfModule mod_mem_cache.c>
  MCacheEnable mem /
  MCacheSize 65000
  MCacheMaxObjectCount 6500
  MCacheMinObjectSize 0
  MCacheMaxObjectSize 580000
 </IfModule>
</IfModule>
```

Making sure a file is cached

How can you tell if your cache is working correctly? First, you should be able to measure performance improvement. If you need to see details, you can set LogLevel to debug and make at least two requests for the same file to your Apache Web server.

Example 9-3 displays the error log trace of two requests performed on the default Apache Web page (index.html, which includes apache_pg.gif). The first request services the two files from the disk and caches only the GIF file, because the no-cache parameter was in the request and the CacheIgnoreCacheControl was not set to On. As you can see, the second request will handle the GIF file from the cache.

You should also monitor your memory utilization; with caching on, it should increase.

Example 9-3 Cache module logging

```
[debug] mod_cache.c(109): cache: URL / is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html.var is being handled by mem
[debug] mod_cache.c(109): cache: URL /apache_pb.gif is being handled by mem
[debug] mod_cache.c(194): cache: no cache - add cache_in filter and DECLINE
[debug] mod_cache.c(419): cache: running CACHE_IN filter
[debug] mod_cache.c(650): cache: Caching url: /apache_pb.gif
[debug] mod_cache.c(681): cache: Added date header
[debug] mod_cache.c(109): cache: URL / is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html is being handled by mem
[debug] mod_cache.c(109): cache: URL /index.html.var is being handled by mem
[debug] mod_cache.c(109): cache: URL /apache_pb.gif is being handled by mem
[debug] mod_cache.c(211): cache: fresh cache - add cache_out filter and handle request
[debug] mod_cache.c(339): cache: running CACHE_OUT filter
[debug] mod_cache.c(351): cache: serving cached version of /apache_pb.gif
```

9.3.5 Optimize the network subsystem

Network subsystem optimization has several key tasks:

- Remove useless network traffic and connections.
- Remove time consuming network operations.
- Compress the data during transfer to save bandwidth.

The following is the directives that you should check:

- HostnameLookups

  This directive is set to *off* by default in V2.0 and V1.3, but was on by default for earlier versions. Setting this directive to on will give you the DNS name of the browser performing a request to your server instead of using only the IP address. In addition to generating network traffic, setting this directive to on will add latency, since reverse DNS lookup is usually slow.

  ```
  HostnameLookups off
  ```

If you set this directive to off, a log entry will look like this:

```
137.65.67.59 - - [24/May/2002:09:38:16 -0600 "GET /apache_pb.gif
HTTP/1.1" 200 2326
```

If you set this directive to on, the log entry will look like this:

```
furby.provo.novell.com - - [24/May/2002:09:37:54 -0600 "GET
/apache_pb.gif HTTP/1.1" 200 2326
```

If your log analysis requires the resolution of IP addresses, consider using a tool such as **logresolve** to perform this translation. You can find this tool at:

http://httpd.apache.org/docs/2.0/programs/logresolve.html

- Timeout

  This directive is set, by default, to 300 seconds. This is the maximum delay Apache allows an HTTP connection to remain open after the last interaction. This value is excessively high because no user will ever wait five minutes to complete a request. Our recommendation is to reduce it until valid connections do not time out:

  ```
  Timeout 60
  ```

- KeepAlive

  The default value is on and we recommend that you keep it this way.

  ```
  KeepAlive on
  ```

  This parameter allows for persistent connections and multiple sequential requests from the client inside the same TCP connection, allowing a much faster dialog between the client and the server. In addition to being faster, KeepAlive reduces traffic on the link by removing the connection negotiation for each request.

  KeepAlive will provide your users with a huge performance improvement by decreasing the latency between requests. The latency could be cut by two thirds if your server is not overloaded. With a server using most of its CPU capacity, your gain will be twofold:

  – Serving more clients in the same time interval because more CPU cycles are available and network bandwidth is less utilized.

  – Faster response to your users.

  For a loaded server, you should easily see a gain of around 50% in the number of requests per second.

- MaxKeepAliveRequests

  The default value is usually equal to 100. This value limits the number of HTTP requests for which a single TCP connection will stay alive. Persistent connections will automatically be ended after that value is reached and connection negotiation will need to be restarted after this point.

  A high value is preferable, but needs to be set in conjunction with the KeepAliveTimeOut parameter to be able to clean up the dead connections at regular intervals. This value could also be set to 0, allowing an unlimited number of requests within a single connection.

  We recommend setting this value to as high a number as possible, especially if your users are habitually requesting many files in the same session.

  ```
  MaxKeepAliveRequests 400
  ```

  When this parameter reaches its limit, the TCP connection will terminate and will need to be re-initiated from the client browser.

- KeepAliveTimeOut

  This parameter sets the maximum time Apache will wait between two requests before ending the connection. The default value is 15 seconds. Whether to change this value or not depends on your network speed and traffic. If multiple browsers are not properly closing the KeepAlive connections with the server, these connections will stay unavailable for other clients during this period. On the other hand, for a slow connection (a modem, for example), the timeout value may need to be increased or each request will have to go through the connection process again and again during a single session.

  ```
  KeepAliveTimeOut 15
  ```

  When this parameter reaches its limit, the TCP connection will terminate and will need to be re-initiated from the client browser.

- Use of sendfile

  Apache normally uses sendfile to supply static files to the Web client. However, if the files are stored on an NFS file system, the files are not cached by Apache, so performance may suffer. If using NFS, consider disabling the directive.

  ```
  EnableSendfile Off
  ```

Tip: Use local disks for Web content wherever possible. Try not to use NFS in conjunction with Apache, because the access to NFS shares will increase the latency of your Web server.

Compression of data

Compression of text files can reduce their sizes significantly, thereby reducing networking costs and improving performance. For example, reductions of 72% are possible using compression level 6 (medium). However, compression requires CPU capacity; the higher the level of compression (values 6-9 are valid), the greater the CPU requirements, so there is a trade-off.

Apache can compress the following files using GZIP-encoding:

- HTML files, text files, and any other text files
- Postscript files

Some files should not be compressed. These include:

- Any files that are already compressed, like JPEG/GIF/PNG images, zip/gz/bz2 archives, RPM packages, PDF documents, and so on
- Javascript and CSS files, mainly because of bugs in browser software

Apache uses module mod_deflate to perform the compression. See Figure 9-9 for an example of an HTML file with and without compression enabled.

- Web server: Apache/2.0.46
- Document Path: /compress_redpaper.html
- Concurrency Level: 150
- Complete requests: 50000
- Failed requests: 0
- Broken pipe errors: 0

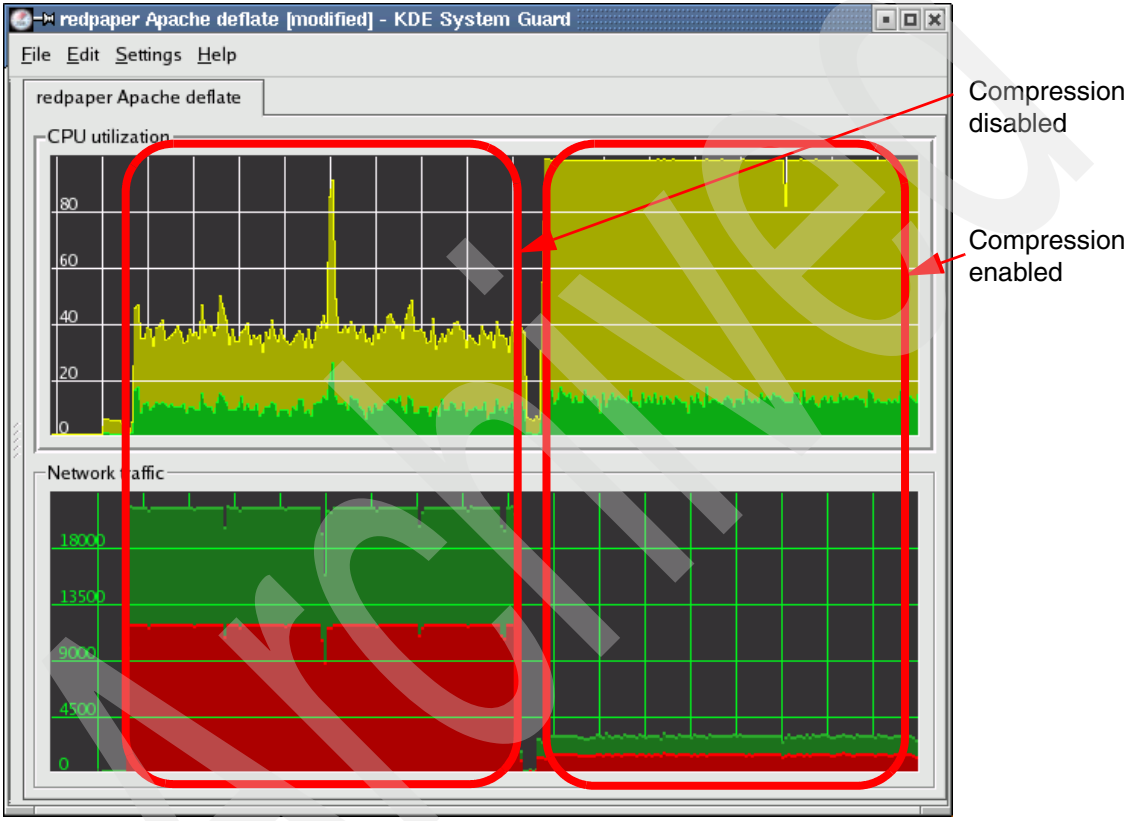


Figure 9-9 Apache 2 without data compression (left) and using data compression (right)

Analyzing the data and graphs, the following can be seen:

- Compression reduced network bandwidth by 70%.
- Compression increased CPU utilization by 87% or more. The CPU was saturated in our tests.
- With compression enabled, only one third of the number of client requests could be serviced.

You will need to determine which is the best option to choose for your configuration based on client needs and associated server hardware requirements.

What should be compressed

You can specify which file types are to be compressed by a directive in the httpd.conf file:

```
#Compression only HTML files
AddOutputFilterByType DEFLATE text/html text/plain text/xml
```

For all types of files (except images file), the compression will be done where we put the filter DEFLATE, such as:

```
<Location />
SetOutputFilter DEFLATE
```

Some browsers and versions of browsers have problems with the compression of specific types of files. To filter what each one should receive, use the directive **BrowserMatch**:

```
#Netscape 4.x
BrowserMatch ^Mozilla/4 gzip-only-text/html
#Netscape 4.06-4.08
BrowserMatch ^Mozilla/4\.0[678] no-gzip
#MSIE working like as Netscape
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
```

For images that do not need to be compressed, such as JPG, GIF, and PNG, compression can be disabled with these directives:

```
#Exception for images already compressed
SetEnvIfNoCase Request_URI \
\.(?:gif|jpe?g|png)$ no-gzip dont-vary
```

The **vary** directive is used to advise the proxies to send only compressed contents to clients that understand it:

```
#Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary
</Location>
```

Important: Apache 2 only compresses data prior to sending it to the client if the HTTP header of the client's request includes either:

```
Accept-encoding: gzip
Accept-encoding: gzip, deflate
```

Compression directives

These directives determine the characteristics of the mod_deflate module and its impact on server performance.

For details of all directives in mod_deflate, see:

http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

- DeflateBufferSize

  Specifies the blocks of memory that should be compressed at one time. The default is 8192.

  ```
  DeflateBufferSize 16384
  ```

- DeflateCompressionLevel

  Sets the level of compression Apache should be used. The default is 6. Level 9 specifies maximum compression, but at great CPU cost.

  ```
  DeflateCompressionLevel 9
  ```

- DeflateMemLevel

  Defines how much memory Apache should use for compression. The default is 9.

  ```
  DeflateMemLevel 9
  ```

- DeflateWindowSize

  Specifies the zlib compression window size. Generally, the higher the window size, the higher can the compression ratio be expected. The default is 15.

  ```
  DeflateWindowSize 15
  ```

9.4 Further information

Apache is a proven Web server software with a rich function set, high performance, and supports many platforms. It is also the most popular Web server according to the data from Netcraft.com.[2] But as with any other computer software, it has inherent limitations that you will likely encounter. In this situation, you have several options.

Optimize the PHP/Perl/Python/SQL code

For many sites serving mostly dynamic content, the Web server sometimes is not the major resource consumer. If the CPU of your server is busy and the utilization grows very fast when client connections increase, and you have tried the tuning option discussed in this chapter, then the bottleneck may be in your Web application. It may be caused by inefficient PHP script, or slow SQL code, or a badly designed database, and so on, in which case you should optimize the code.

Use load balancing technology

A single server has its hardware limitations. When a subsystem has been the bottleneck and you have exhausted all optimization attempts, then you can try load balancing.

We mention load balancing in 2.1.2, “Scalability” on page 9. You can have two or more IBM System p servers running Apache servers, coordinated by a load balancing software to distribute the client request to these servers, thus overcoming the natural limitation of a certain hardware configuration.

Migrate to IBM HTTP Server

The IBM HTTP Server (IHS) is a powerful, robust, secure, and free Web server based on the popular Apache Web server. IHS takes the latest stable Apache code tree, adds modules to improve performance, security, and usability, and packages it for easy installation.

The IBM HTTP Server product information Web site is:

http://www-306.ibm.com/software/webservers/httpservers/

It is a free download.

[2] Data from:
http://news.netcraft.com/archives/2006/09/05/september_2006_web_server_survey.html

Figure 9-10 shows the comparison between Apache prefork, worker, and IBM HTTP Server. While serving a bit higher page throughput, the IBM HTTP Server consumes lower CPU and memory resources. We believe that after proper tuning, Apache may have similar performance, but if you need a optimized and tuned Web server just out of the box that is compatible with Apache, then try IBM HTTP Server.

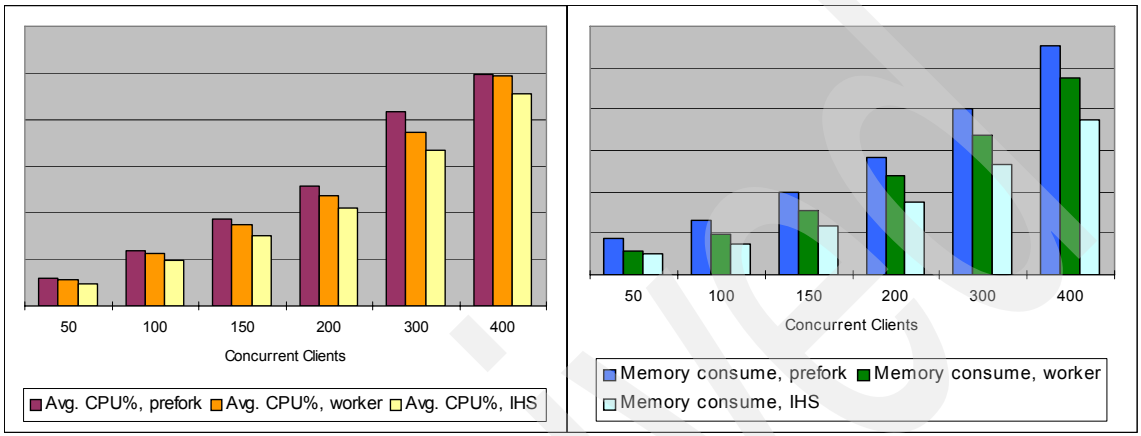


Figure 9-10 The CPU and memory consume compare between Apache prefork, worker, and IBM HTTP Server

Note: The performance data is specific to the testing environment used by the author, including the hardware configuration, software version, and configuration, and the testing workload that has been used. It does not imply that you will have the same result here.

Try other HTTP server software

There are many other HTTP server software products, such as *Lighttpd*. It provides the most commonly used features, such as virtual host, CGI/FastCGI/SSI, scripting language like PHP support, and URL rewrite, and it is reported to have better performance and consumes less resources than Apache. It is available from:

`http://www.lighttpd.net/`

Some really large Web sites are running lighttpd, such as Sourceforge.net. A list can be found here:

`http://trac.lighttpd.net/trac/wiki/PoweredByLighttpd`

10

Oracle Database

In this chapter, we discuss the Oracle Database and performance tips.

Oracle Database 10g Release 2 is the officially supported version for Linux on POWER platform. In this chapter, we use the short term “Oracle” or “Oracle DB” to refer to “Oracle Database 10g Release 2 for Linux on POWER”.

10.1 Oracle architecture

Oracle Database is a quite complex software system, and to be able to tune the performance, we must first understand its architecture. The database is presented in three parts:

- The memory space
- The Oracle processes
- The DBMS files

These are depicted in Figure 10-1.

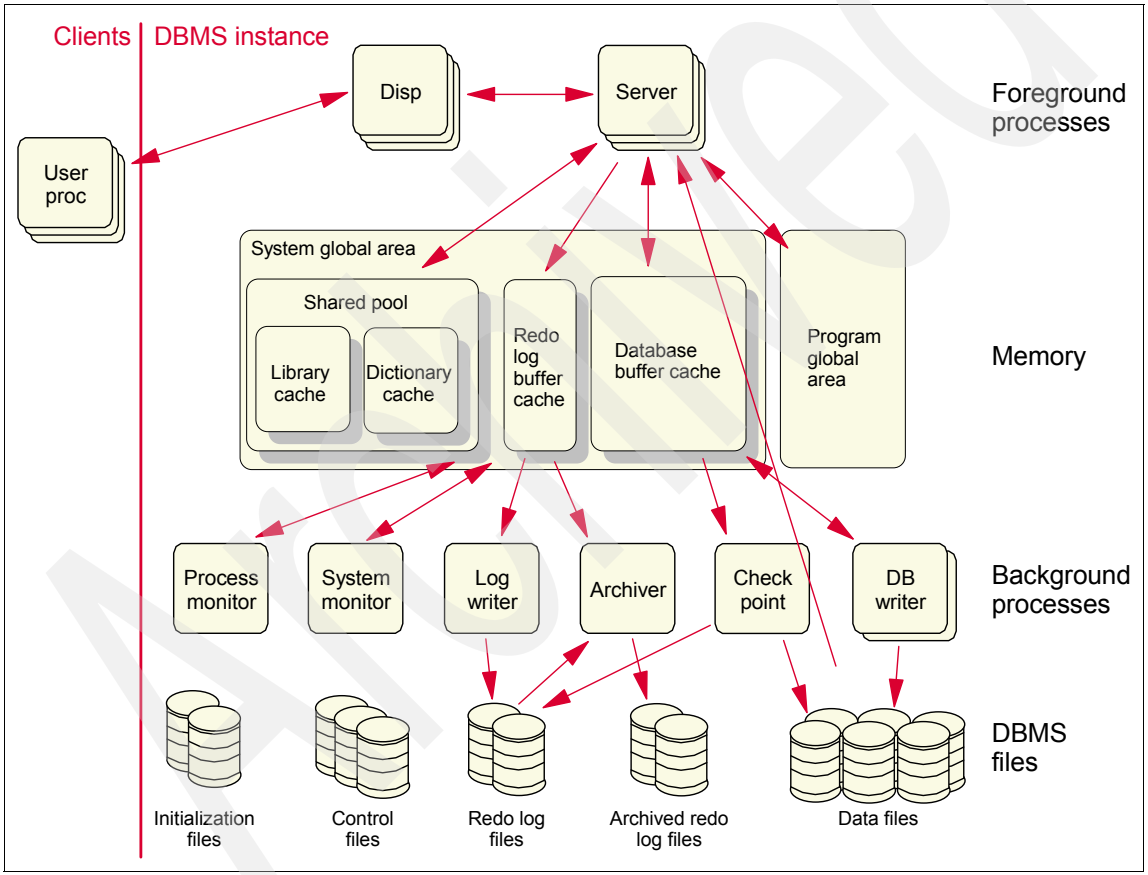


Figure 10-1 Oracle Database structure

10.1.1 Memory architecture

The memory used by an Oracle instance is composed of two main areas: the Program Global Area (PGA) and the System Global Area (SGA).

PGA

Program Global Area is a memory region that contains data and control information for a Oracle server process. It is a non-shared memory created by Oracle when a server process is started. Access to it is exclusive to that server process and is read and written only by Oracle code acting on behalf of it. The PGA can be classified into the following areas:

- Private SQL Area
- Cursors and SQL Areas
- Session Memory
- SQL Work Areas

SGA

System Global Area is a group of shared memory structures that contain data and control information for the Oracle Database instance. An SGA and Oracle processes constitute an Oracle instance, and each instance has its own SGA. For all the users that connect to the same instance, the data in the instance's SGA is shared among the users.

The SGA contains the following data structures:

- Database buffer cache

  The database buffer cache is the cache area of database files in the Oracle SGA. The server process only writes to the database buffer cache. An asynchronous background process (the database writer) asynchronously updates the data files.

- Redo log buffer cache

  The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database, by INSERT, UPDATE, DELETE, CREATE, ALTER, or DROP operations. Redo entries are used for database recovery, if necessary.

  The server processes only write to the redo log buffer cache. When the SQL Commit statement is used, the log writer process then writes the content of the cache to the redo log files.

- Shared pool

  The shared pool memory area is mainly composed of two caches: the Library cache and the Dictionary cache.

  - Library cache

    The Library cache stores the information about the most recently used SQL and PL/SQL statements. Statements that have been ran previously are stored here, so if the same statement is requested to run again, Oracle can just pick the parsed form and run it directly to save time.

  - Dictionary cache

    The Dictionary cache contains the information of the database structure, including tables, indexes, columns, data files, users, and many other object information that describe the database. Since Oracle need this information every time when accessing the actual data, this cache will help Oracle save the time.

- Java pool

  Java pool memory is used in server memory for all session-specific Java code and data within the JVM™.

- Large pool (optional)

Starting with release 9*i*, Oracle can re-size the SGA without shutting down the instance. The same holds for the size of the buffer cache and the shared pool. Another important feature introduced with release 9*i* is the PGA automatic memory management for instances working in dedicated mode.

For more details, please refer to Chapter 8, “Memory Architecture”, of *Oracle Database Concepts 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14220/memory.htm#i12483

10.1.2 Oracle processes

In a Oracle application system, the processes can be categorized into two major groups:

- User processes run the application program (such as a Pro*C program) or Oracle tool code (such as Enterprise Manager or SQL*Plus).
- Oracle processes run the Oracle Database server code. They include Server processes and Background processes.

Oracle processes are the key part of the Oracle Database, since they perform the data processing and managing tasks.

Server processes

Server processes are used to act on behalf of clients, because access to the SGA is allowed for Server processes only and not to User processes. The Server processes handle the requests of User processes connected to the instance, and perform one or more of the following actions:

- Parse and run SQL statements.
- Read necessary data blocks from datafiles on disk into the shared database buffers of the SGA, if the blocks are not already present in the SGA.
- Return results to the User processes.

Two architectures are possible for Server processes:

- Dedicated server mode

  For each User process, a dedicated Server process is created to handle the request.

- Shared server mode

  Many User processes share a small number of Server processes. When a User process requests a connection, it connects to a dispatcher process. The dispatcher process puts the request in the Server processes queue. When a Server process becomes free, the request is taken from the queue and executed.

Background processes

The Oracle background processes include:

- Database writers (DBW*n*)

  These processes write modified (dirty) database blocks from the SGA database buffer cache to the database files. Although one database writer is usually enough (DBW*0*), up to twenty database writers may be activated.

- Log writer (LGWR)

  The log writer writes sequentially batches of modified entries in the SGA's redo buffer for one or more transactions to the online redo logs files.

- Checkpoint (CKPT)

  When a checkpoint occurs, the checkpoint process updates the header of the datafiles to record the detail of a checkpoint. It never performs the buffer to disk write operation; instead, it notifies the database writer process(es) that updates to the data and control files must be completed to the physical files.

- System monitor (SMON)

  The SMON process takes care of instance recovery in the case of a system crash.

- Process monitor (PMON)

  The PMON process performs recovery when a user process fails. It is responsible for cleaning up the database buffer cache and freeing resources that the User process was using. PMON also periodically checks the status of dispatcher and Server processes, and restarts any that have accidentally stopped. PMON also registers information about the instance and dispatcher processes with the network listener.

- Recovery process (RECO)

  The RECO process recovers transactions from a distributed database configuration.

- Job queue processes

  Job queue processes are used for batch processing. They can be viewed as a scheduler service that schedule jobs as PL/SQL statements or procedures on an Oracle instance.

- Archiver (ARC*n*)

  The archiver processes copy the online redo log files to the archive log files. While a single archiver process (ARC*0*) is usually adequate, it is possible to activate up to ten archiver processes. The ARC*n* process is active only if the database works in ARCHIVELOG mode and automatic archiving is enabled.

- Queue Monitor (QMN*n*)

  The QMN*n* processes are optional processes monitoring message queues for Oracle Streams Advanced Queueing.

- Other background processes

  There are several other background processes that may be running, such as MMON, MMNL, MMAN, RBAL, OSMB, and so on.

For more details, please refer to Chapter 9 “Process Architecture”, of *Oracle Database Concepts 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14220/process.htm#i21263

10.1.3 Database files

The third component of the Oracle architecture includes the actual physical files associated with a database instance. These are the files that hold the data, the log, the configuration, and the diagnostic log.

- Data files

  Database data files contain the data, like tables and indexes.

- Control files

  The control file includes information pertinent to the database itself. It contains such information as the name of the database, date and time stamp of creation, location of database files and redo logs, and synchronization information. Without this file, the database will not start properly.

- Redo logs

  The redo log files are the “journal” of the Oracle Database. All changes made to the user or system objects are recorded through the redo logs. The redo logs are also used for error recovery in case of media failure or system crash.

- Archive log files

  Archive log files are the archive of the Redo logs.

- Parameter files

  Parameter files contain a list of parameters for a instance and database. It is a small file that contains the configuration of the Oracle Database. The Oracle options we tuned are written into this file.

- Alert and trace log files

  These are the files where the server and background processes write their error logs. Some of the information written to a trace file are intended for the database administrator, while other information is for Oracle Support Services. Trace file information is also used to tune applications and instances.

10.2 Potential bottlenecks

The Oracle Database application system is usually a very complex system; the client application that make use of database system often need to implement some complicated business logical flow, or store a large amount of data and a running analyses procedure, or support hundreds to thousands of clients to view, add, modify, or delete data. To support various kind of applications, the database theory and software has become so complex that sometimes it is even considered more complex than the operating system.

So the bottleneck of such a complex system will not be a simple fixed subsystem; in a different application environment, it will likely be a different possibility. Let us first look at the typical workload type of a database.

10.2.1 Database workloads

A database is a self-described collection of data managed by a software product (database management system (DBMS)) for a specific purpose. Among the different possible uses of a database, two are particularly relevant from both a commercial and architectural point of view:

- Transactions recording, known as *online transaction processing* (OLTP)
- Decision support, known as *online analytical processing* (OLAP)

While the data structures are quite often the same (that is, they are both relational databases), the technical structure of an OLTP system is very different from that of an OLAP one.

Online transaction processing

OLTP systems are transaction recording systems. The term *Transaction* here usually has two meanings. One is the business or commercial transaction and the other is the computer technology meaning. A transaction in database technology is an atomic group of data processing steps, which means that either all the steps of the transaction completed successfully, and the result has been committed, or if any single step failed to execute, the whole group must be rolled back and the data are returned to the original state of when the transaction began.

Typical operations in an OLTP database are modifying the data records or storing new records. They have usually been read, processed, modified, and stored. Oracle Database needs to keep a log of all the operations, so failed transactions can be rolled back, and in case of any system failure, the database can recover using the latest data and the log.

The database of a transaction system usually consists of many relatively small tables. “Many” can be thousands or more for some large application systems, for example, SAP R/3® can have 10,000-20,000 tables. And “relatively small” means compared with the OLAP database, although they can be hundreds of gigabytes large, in order to store the transaction data.

Online analytical processing

OLAP systems are analytical systems, meaning that the typical user activity is not data entry but data analysis. Hence, data integrity is not as important as it is for OLTP systems. What is important is providing the users with an efficient and flexible tool for in-depth data manipulation.

The OLAP acronym was introduced in *Providing OLAP to User-Analysts: An IT mandate* by Codd, et al. According to the authors, an OLAP software should have the following twelve properties:

- Multidimensional conceptual view
- Transparency and accessibility
- Consistent reporting performance
- Client/server architecture
- Generic dimensionality
- Dynamic sparse matrix handling
- Multi-user support
- Unrestricted cross-dimensional operations
- Intuitive data manipulation
- Flexible reporting
- Unlimited dimensions
- Aggregation levels

The intrinsic nature of OLAP operations is multidimensional. Queries such as “I want to know how many System p servers have been sold in Texas in April” are three-dimensional queries whose dimensions are products (System p servers), state (Texas), and time (April). Relational databases have not been built for multidimensional queries. These queries need many joins, while joins of more than five tables typically need too much time.

From many points of view, the best DBMS technology for OLAP is that of *multidimensional* databases (MDDBs), such as Oracle Express. Data is recorded in an operational system (an Oracle relational database OLTP system) and then transferred to the multidimensional database (Oracle Express). Satisfactory performance can be obtained by performing analysis directly on an MDDB, but the data is only as recent as the latest data loaded from the operational system. A more flexible solution is to move operational system data to a new relational database tuned for OLAP activity, the *data warehouse* or *data mart*.

Data warehouses typically consist of a very large table (fact table) and some smaller tables (dimension tables). Typical operations are joins between the fact table and the dimension tables.

User activity consists of few operations reading large amounts of data. It is a mistake, however, to tune the system only for reading activity. During data loading, data must be written in Oracle data files, indexes must be changed, and sometimes logs must be recorded. If no time constraints exist for loading, it is acceptable to tune the system for pure reading, but if large amounts of data must be loaded often in a short time, it is important to design the system so that I/O is also efficient during writing.

10.2.2 Important subsystems

The Oracle Database server requires a lot of CPU power and large amounts of memory to maximize the performance of applications that are computation intensive. An efficient disk subsystem is also very important, especially when handling many random I/O requests. Its primary function is to search and retrieve data from the disk subsystem as requested by client queries. The important subsystems to look at are:

- Processor
- Memory
- Disk subsystem

Regarding the two workload type we discussed in 10.2.1, “Database workloads” on page 306, they have different characteristics and thus have different requirements. An OLTP system often has many users connected, processing transaction data and storing it, so it demands a faster processor and write efficient disk system. An OLAP system is mostly running read operations to retrieve data from disks, analyzes the data, and generates the report, so a read efficient disk subsystem and large memory for caching is important.

10.2.3 Monitoring Oracle Database

For a generic operating system level monitoring tool, please refer to Chapter 5, “Linux monitoring tools” on page 87.

Let us take a look at the facilities provided by Oracle for assist with database monitoring.

$V performance views

Oracle has a set of database views called dynamic performance views, since their contents are primarily real time performance data. These views are maintained by the database server, reflect the internal data of an Oracle instance, such as memory structure and disk structure, and is continuously updated by the server while the database is open and in use. The data of these views can be accessed by the Oracle user “SYS” or anyone with the “SYSDBA” role, like any other tables, through the SELECT SQL clause.

These views all have public synonyms with the prefix $V, so often referred as “$V views”. The Oracle Enterprise Manager also use these $V views for performance data collection. Actually, most of the Oracle performance monitoring tools read the data from these $V views and sort them into human readable reports.

Here is a list of useful views:

- System information
  - V$OSSTAT

    System utilization statistics from the operating system.
  - V$SYSSTAT

    Lists system statistics.
- Database information
  - V$DATABASE

    Information about the database from the control file.
  - V$INSTANCE

    The state of the current instance.
  - V$PROCESS

    Information about the currently active Oracle processes.
  - V$SESSION

    Session information for each current session.
  - V$SESSION_WAIT

    The resources or events for which active sessions are waiting.
  - V$SEGMENT_STATISTICS

    Information about segment-level statistics.
  - V$UNDOSTAT

    Undo space consumption, transaction concurrency, and length of queries executed in the instance.
  - V$WAITSTAT

    Lists block contention statistics.
- Memory information
  - V$SGA

    Summary information about the SGA.
  - V$SGASTAT

    Displays detailed information about the SGA.
  - V$SGA_TARGET_ADVICE

    Information about the `SGA_TARGET` initialization parameter.

- V$PGASTAT

  PGA memory usage statistics and statistics about the automatic PGA memory manager if it is enabled.

- V$PGA_TARGET_ADVICE

  Predicts how the cache hit percentage and over allocation count statistics displayed by the V$PGASTAT performance view would be impacted if the value of the PGA_AGGREGATE_TARGET parameter is changed.

- V$BUFFER_POOL

  Information about all buffer pools available for the instance.

- V$BUFFER_POOL_STATISTICS

  Statistics about all buffer pools available for the instance.

- V$SHARED_POOL_ADVICE

  Information about estimated parse time in the shared pool for different pool sizes.

- V$DB_CACHE_ADVICE

  Predicts the number of physical reads for the cache size corresponding to each row.

- I/O information
  - V$ASM_DISK & V$ASM_DISK_STAT

    The ASM disk statistics, including the number of read and write operations and the wait time.

  - V$FILESTAT

    The number of physical reads and writes done and the total number of single-block and multiblock I/Os done at the file level.

  - V$SESS_IO

    I/O statistics for each user session.

Many of these views depend on following the initialization parameters:

- STATISTICS_LEVEL

  Specifies the level of collection for database and operating system statistics. It should be set to TYPICAL for Oracle to collect all the major statistics.

- TIMED_STATISTICS

  Specifies whether or not statistics related to time are collected. Set it to TRUE to make Oracle collect timed statistics and access it from the V$ views.

This is a very limited list. For a complete list of V$ views and the information they can provide, please refer to Part III, “Dynamic Performance Views”, in *Oracle Database Reference 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14237/dynviews_part.htm#i403961

Also refer to Chapter 10, “Instance Tuning Using Performance Views”, in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/instance_tune.htm#i35312

Although the dynamic performance views contain useful performance data, it is not a straight forward process to select the useful data and sort it into an easily readable format. Instead, we can use tools like Statspack or AWR to ease the task.

Statspack

Statspack is a set of performance monitoring and reporting tools provided by Oracle. It is available from Oracle 8i and often considered the successor of the BSTAT/ESTAT scripts in the former version of Oracle Database. The tool is a set of SQL, PL/SQL, and SQL*Plus scripts that allow the collection, automation, storage, and viewing of performance data associated with the instance. These scripts collect the performance statistics data and store them permanently in database tables, which can later be used for reporting and analysis.

The basic idea of statspack is snapshots and reports. It allows a DBA to have a series of snapshots that represents the database status at a certain point in time, and reports can be generated from two snapshots, that is, the statistics data in the time period enclosed by the two snapshots.

Statspack comes with Oracle, but needs manual installation. Please follow the instructions in $ORACLE_HOME/rdbms/admin/spdoc.txt for installing, configuring, and using the tool.

Using statspack is straight forward: Create a snapshot, let your database run, create another snapshot, and then create a report from the two snapshots.

Connect to the database as the perfstat user and create a snapshot.

```
SQL>  connect perfstat/perfstat_password
SQL>  execute statspack.snap;
```

Run your database application, take snapshots during the run, and then create the report by running the spreport.sql script. The script will ask for the beginning

and ending snapshot ID, generate the report, and ask for the file name to save the report:

```
SQL>  @?/rdbms/admin/spreport
```

Example 10-1 shows part of a statspack report (a report is usually near two thousands lines long).

Example 10-1 Part of the statspack report

```
STATSPACK report for

Database    DB Id    Instance     Inst Num Startup Time    Release     RAC
~~~~~~~~~ ---------- ------------ -------- --------------- ----------- ---
          3920964862 ora10g              1 29-Sep-06 18:53 10.2.0.2.0  NO

Host  Name:   k-rhel.itsc.aust Num CPUs:    4        Phys Memory (MB):       1
~~~~

Snapshot       Snap Id     Snap Time      Sessions Curs/Sess Comment
~~~~~~~~~    ---------- ------------------ -------- --------- -------------------
Begin Snap:         11 29-Sep-06 18:55:01       68      22.9
  End Snap:         12 29-Sep-06 19:05:48       18       4.9
   Elapsed:               10.78 (mins)

Cache Sizes                       Begin        End
~~~~~~~~~~~~~                  ---------- ----------
               Buffer Cache:       752M       768M  Std Block Size:         8K
           Shared Pool Size:       192M       176M     Log Buffer:    13,920K

Load Profile                            Per Second       Per Transaction
~~~~~~~~~~~~                            ---------------  ---------------
                  Redo size:            360,361.01               573.73
              Logical reads:            176,835.37               281.54
              Block changes:              2,699.70                 4.30
             Physical reads:                369.34                 0.59
            Physical writes:                125.02                 0.20
                 User calls:              1,161.32                 1.85
                     Parses:                592.25                 0.94
                Hard parses:                  0.87                 0.00
                      Sorts:                 76.43                 0.12
                     Logons:                  0.02                 0.00
                   Executes:              1,878.83                 2.99
               Transactions:                628.10

  % Blocks changed per Read:    1.53    Recursive Call %:    79.15
```

```
 Rollback per transaction %:   65.19       Rows per Sort:   988.65

Instance Efficiency Percentages

            Buffer Nowait %:  100.00       Redo NoWait %:   99.98
            Buffer  Hit   %:   99.79    In-memory Sort %:  100.00
            Library Hit   %:  100.09        Soft Parse %:   99.85
         Execute to Parse %:   68.48         Latch Hit %:   99.70
Parse CPU to Parse Elapsd %:   57.29     % Non-Parse CPU:   99.27

 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   68.80   67.68
    % SQL with executions>1:   60.04   93.10
  % Memory for SQL w/exec>1:   58.96   96.03
```

As of the Oracle 10g release, a new tool named Automatic Workload Repository (AWR) was provided and replaced statspack, but statspack is still an important tool for Oracle monitoring and tuning, and extremely useful information can be collected by a few simple steps.

For more information, please refer to Chapter 21, “Using Statspack”, in *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*, found at:

http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96533/statspac.htm

Automatic Workload Repository

The Automatic Workload Repository (AWR) is an integrated part of the Oracle server. Its purpose is to collect server-related performance data periodically (every 60 minutes by default), when the statistics_level parameter is set to typical or all. You can view the captured data in the AWR report, or let the Automated Database Diagnostic Monitor (ADDM) component of the server use this data to diagnose performance issues. The performance recommendations can be accessed by using Oracle Enterprise Manager.

The idea of AWR is quite similar to statspack: Have a series of snapshots and generate reports from snapshots. Unlike statspack, AWR is installed by default and ready to run in Oracle 10gR2 (the default snapshot period is one hour).

We have two ways to access the AWR function: through the Oracle Enterprise Manager Web interface, or by running commands in SQL*Plus.

Figure 10-2 shows the entrance for AWR in the EM Web interface.

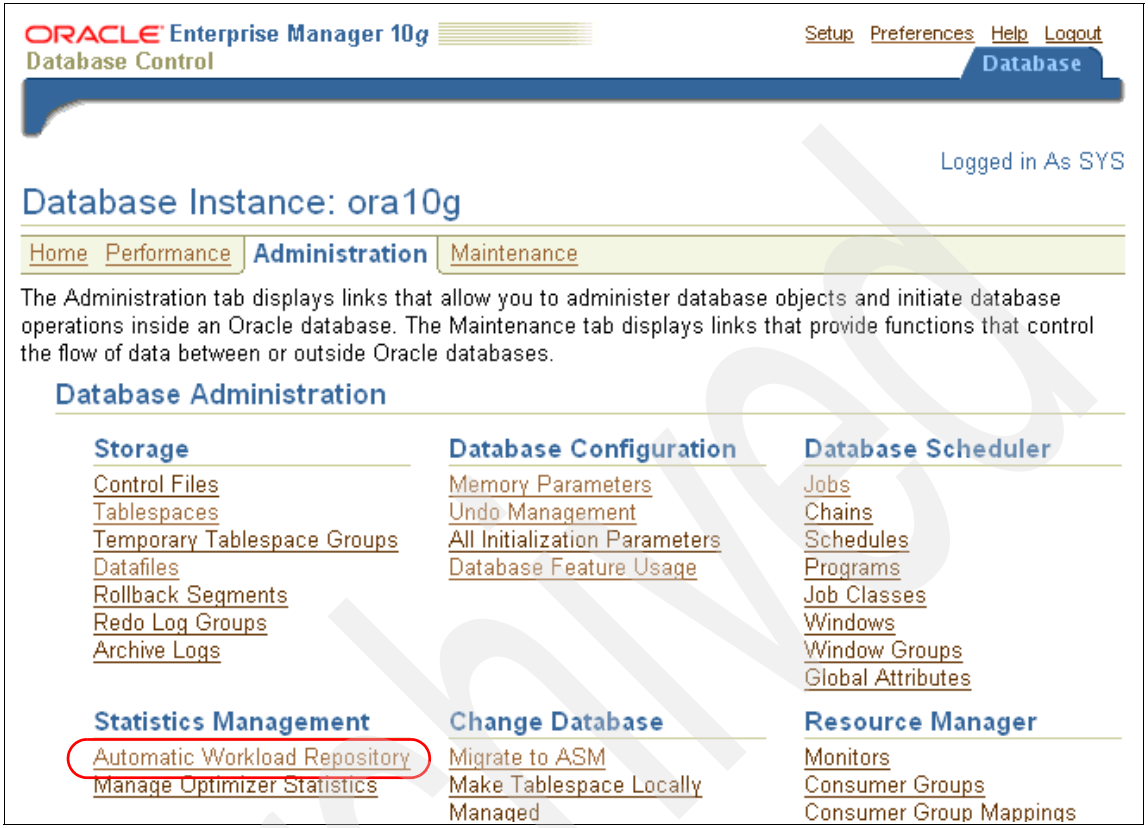


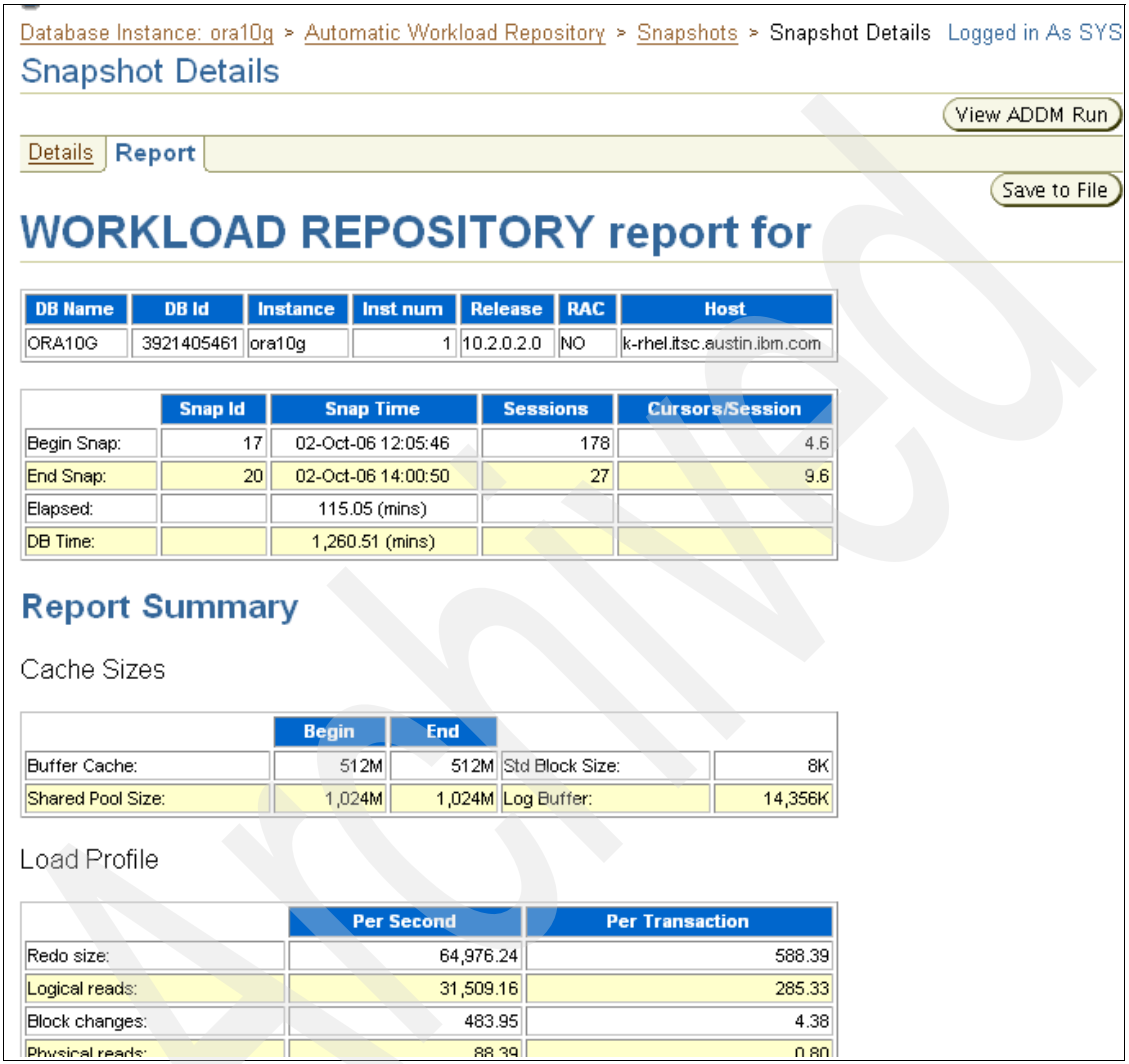
Figure 10-2 Access AWR in Oracle EM web interface

In the Web interface, you could set up the snapshot interval, collection level, take a snapshot manually, exam the snapshots and reports, and so on.

You can also create a snapshot from the SQL*Plus command line:

```
SQL> connect sys as sysdba
SQL> execute dbms_workload_repository.create_snapshot
```

As in statspack, a report can be generated from two snapshots. Figure 10-4 on page 317 is the AWR generated report in HTML format.



Database Instance: ora10g > Automatic Workload Repository > Snapshots > Snapshot Details Logged in As SYS

Snapshot Details

View ADDM Run

Details | Report

Save to File

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORA10G	3921405461	ora10g	1	10.2.0.2.0	NO	k-rhel.itsc.austin.ibm.com

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	17	02-Oct-06 12:05:46	178	4.6
End Snap:	20	02-Oct-06 14:00:50	27	9.6
Elapsed:		115.05 (mins)		
DB Time:		1,260.51 (mins)		

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	512M	512M	Std Block Size:	8K
Shared Pool Size:	1,024M	1,024M	Log Buffer:	14,356K

Load Profile

	Per Second	Per Transaction
Redo size:	64,976.24	588.39
Logical reads:	31,509.16	285.33
Block changes:	483.95	4.38
Physical reads:	88.39	0.80

Figure 10-3 AWR report

Please refer to following resources for more detail about AWR:

Chapter 5, “Automatic Workload Repository”, in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/autostat.htm#sthref362

Oracle Database 10g: The Top 20 Features for DBAs. Week 6: Automatic Workload Repository, found at:

http://www.oracle.com/technology/pub/articles/10gdba/week6_10gdba.html

Automated Database Diagnostics Monitor

Automated Database Diagnostics Monitor (ADDM) is a component of the database that automatically analyzes the AWR data on a regular basis, then locates the root causes of performance problems, provides recommendations for correcting problems, and identifies non-problem areas of the system. ADDM identifies the most resource-intensive components or operations and provides advice, which may recommend running an advisor or making configuration changes to your database.

An ADDM analysis is performed every time an AWR snapshot is taken and the results are saved in the database. You can view the results of the analysis using Oracle Enterprise Manager or by viewing a report in a SQL*Plus session.

Figure 10-4 shows a ADDM analyzing report in HTML format.

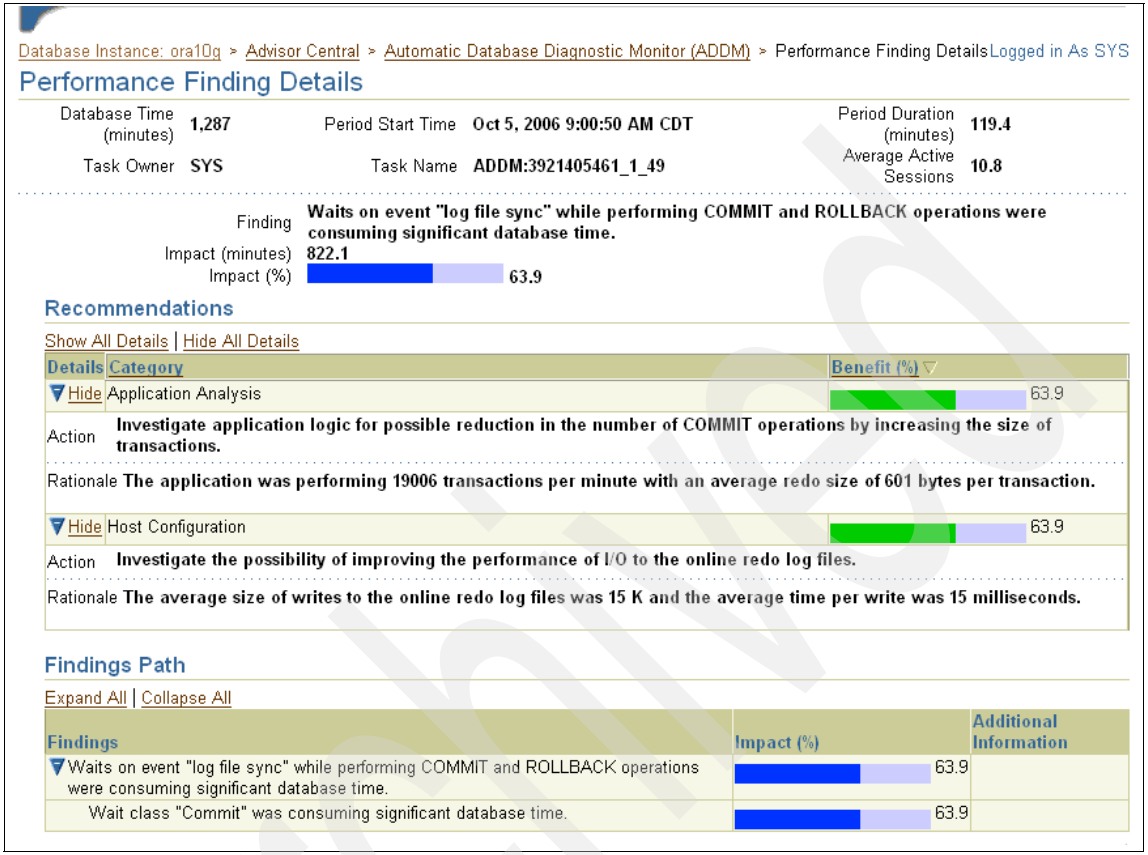


Figure 10-4 ADDM performance analyzing report

Please refer to the following resources for more information:

Chapter 6, “Automatic Performance Diagnostics”, in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/diagnsis.htm#sthref419

Oracle Database 10g: The Top 20 Features for DBAs. Week 18: ADDM and SQL Tuning Advisor, found at:

http://www.oracle.com/technology/pub/articles/10gdba/week18_10gdba.html

Diagnosing and Resolving Performance Problems Using ADDM, found at:

http://www.oracle.com/technology/obe/10gr2_db_single/manage/addm/addm_otn.htm

10.3 Tuning Oracle Database

Tuning an Oracle Database system requires a series of steps to discover the bottleneck and modify the system and Oracle parameters to try to eliminate or minimize the bottleneck. Usually, the work has the following parts:

- Application tuning

  Generally, this is the most important part of the process, and in most cases will give largest performance boost. A database application is usually complex, because you must deal with several elements, such as SQL code, data processing, the network, job schedule, and so on. If the application has a improper architecture, or uses inefficient SQL code, then tuning the database itself will bring very little improvement. Application tuning related to Oracle Database includes, but is not limited to:

  - Database structure design tuning
  - SQL code optimization
  - Database connection pooling and result caching
  - Network connection optimization

  The application tuning is out of the scope of this book; please refer to the following resources:

  - *Part IV Optimizing SQL Statements of Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

    http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/part4.htm#sthref1059

  - *Oracle Database Application Developer's Guide - Fundamentals 10g Release 2 (10.2)*, found at:

    http://download-east.oracle.com/docs/cd/B19306_01/appdev.102/b14251/toc.htm

- Operating environment tuning

  The operating environment includes the server hardware and the operating system. The hardware configurations, such as processor speed, memory capacity, disk type, and quantity, will largely affect the performance behavior of Oracle Database. The operating system is designed to support many kinds of applications and workloads. To make it best fit into the database workload,

it must be tuned. The operating environment tuning includes, but is not limited to:

- Hardware related tuning, like memory capacity, disk RAID level and strip unit, and so on
- Program execution and schedule
- Memory layout optimization
- Disk I/O scheduler and file system tuning
- Network tuning

- Oracle Database tuning

  The goal of database tuning is to make good use of hardware resources and operating system characteristics, ensure a balanced utilization of the resources, and avoid bottlenecks. Tasks include:

  - Memory tuning

    Determine the optimal size of SGA, PGA, buffer cache, shared pool, and so on.

  - Disk I/O tuning

    Choose storage option, database block size, data file distribution, and so on.

  - Resource contention tuning

    Remove process contention, deadlock, and so on.

  Most Oracle tuning is done by modifying the database parameters, mostly by using the ALTER clause.

In this chapter, we will focus on the operating environment and Oracle Database tuning.

10.3.1 Scenario brief

In this chapter, the environment used to run the Oracle Database workload is a partition running Red Hat Enterprise Linux 4, with three processor cores in dedicated mode and a maximum of 30 GB of memory. All the tests in the following sections are running on this partition with fixed processor power; the memory capacity is adjusted according to the test scenario. The disk storage used by the partition is several virtual disks provided by the Virtual I/O server running in another partition, backed by a RAID 5 array of six hard drives.

The tool used to generate the workload is Swingbench, which is a free Java tool that can stress test the Oracle Database 9i and 10g. The tool and the document are available here:

http://www.dominicgiles.com/swingbench.php

10.3.2 General optimization

First, follow the operating system requirement,as described in the Oracle installation document, and make sure the system ulimit and kernel parameters are set properly.

System ulimit

To ensure Oracle processes have enough resources to grow, add these lines to the /etc/security/limits.confs file:

```
oracle soft  nproc   2047
oracle hard  nproc   16384
oracle soft  nofile  1024
oracle hard  nofile  65536
```

These are the recommended values from Oracle; if you are running a really large database and have enough system resources, you can increase the value.

Kernel parameter

Add the following lines to the /etc/sysctl.conf file:

```
kernel.shmall = value must be larger than SGA size
kernel.shmmax = value must be larger than SGA size
kernel.shmmni = 4096
kernel.sem = 250 32000 100 128
fs.file-max = 65536
net.ipv4.ip_local_port_range = 1024 65000
net.core.rmem_default = 262144
net.core.rmem_max = 262144
net.core.wmem_default = 262144
net.core.wmem_max = 262144
```

Note: The two parameters, kernel.shmall and kernel.shmmax, are very important for SGA size. The value of kernel.shmall sets the upper limit of the total shared memory size in the system, in unit of page size, usually 4 KB for POWER5 and POWER5+. And kernel.shmmax sets how large a single shared memory segment can be in units of byte. These two parameters should be set to a larger value than your Oracle SGA size.

For example, you have 32 GB of physical memory and you would like Oracle to use up to 25 GB as SGA, in 4 KB page size units:

25 * 1024 * 1024 / 4 = 6553600

You might want to set the values a little high just to be safe, so set kernel.shmall to around 6600000 and kernel.shmmax to around 27000000000.

10.3.3 Optimize processor subsystem and process execution

As we discussed in 10.1.2, “Oracle processes” on page 302, an Oracle Database system has many processes running that serve the clients and perform operations on the data. Tuning the database to make sure the processes are working properly is the major topic of this section.

SMT

IBM System p servers using POWER5/POWER5+ offer hardware multithreading called SMT. Please refer to “Simultaneous multi-threading (SMT)” on page 37 for detailed information.

The SMT functions enable a database server to serve more concurrent requests and respond more quickly, since it will allow higher thread concurrency. You do not need any extra work to enable SMT; Linux by default has SMT enabled and Oracle also uses the extra hardware thread automatically.

Figure 10-5 shows the performance difference for tuning with and without SMT. The test scenario is a 10 GB Oracle Database with 50/100/150 concurrent connected users running transactions. The result is clear: SMT offers around a 40% performance gain.

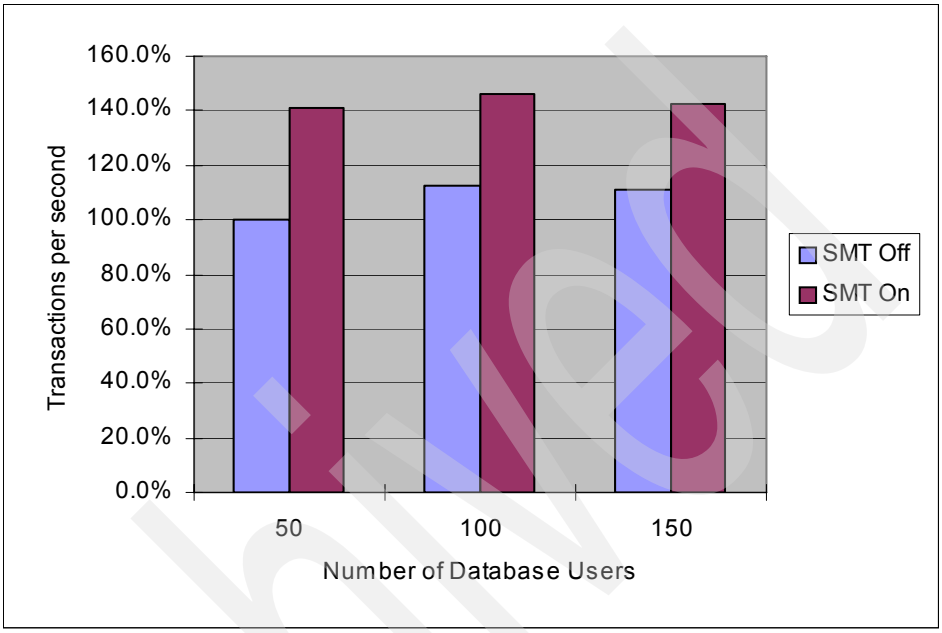


Figure 10-5 SMT effect on database transaction processing

Process/Session upper limit

- PROCESSES

  This initialization parameter is important. It specifies the maximum number of processes that the Oracle instance can start, thus setting the limit of how many user processes can connect to the database. Also, many other parameters are derived from it, so selecting an adequate value is important.

- SESSIONS

  This parameter specifies the maximum sessions that the system supports. Since every user login is a session, this parameter determines the maximum number of users that can connect to the instance. The Oracle default sets it to (1.1 * PROCESSES) + 5. If you want to set it manually, the value should be the number of concurrent users you are going to support, plus the number of background processes, and add approximately 10% uplift to be safe.

Set the two parameters by using the following command in SQL*Plus:

```
SQL> alter system set processes=50;
SQL> alter system set sessions=500;
```

Increase the value to the number of user sessions you want to support.

Dedicated or shared server mode

As we have discussed in 10.1.2, “Oracle processes” on page 302, these two server modes handle user process connections differently. Since dedicated mode has a server process for each connection, while shared mode has a pool of server processes to handle multiple connections, it needs less process capacity to handle same amount of connections, so shared mode server can support more user connections.

A shared server can be beneficial in the following situations:

- Many user connections

  If the database system needs to support many concurrent users connected to perform transactions, then using a shared server can reduce the number of processes and the amount of memory consumed.

- High connection rate

  For shared servers, when a connect request comes in, the dispatchers are already there to handle the connection and will put it into the queue immediately for the server process to pick up. With dedicated servers, a new server process has to be initialized first before accepting the connect request.

However, a dedicated server is good for scenarios where only a small number of connections are made to the database and each of the connections will last for a period of time or even persist to perform a long-running task instead of disconnect and reconnect frequently. Generally, dedicated and shared servers will provide similar transaction processing speeds, but a dedicated server will cause Oracle to fork many processes, so it will consume more memory than a shared server, and the system load will be higher.

Shared server mode is enabled by setting the SHARED_SERVERS initialization parameter to a value greater than zero. It can be set dynamically to enable the shared server at .

Issue following command in SQL*Plus to set the SHARED_SERVERS parameter:

```
SQL> alter system set shared_servers=5;
```

Set SHARED_SERVERS to zero to disable shared servers.

Shared server configuration

There are several parameters related to a shared server and its performance:

- SHARED_SERVERS

  The initial number of shared servers to start and the minimum number to run.

- MAX_SHARED_SERVERS

  The maximum number of shared servers.

- SHARED_SERVER_SESSIONS

  The total number of shared server user sessions that can run concurrently.

- DISPATCHERS

  Configures the dispatcher processes in the shared server architecture. Refer to “Dispatcher configuration” on page 326 for a detailed discussion of this parameter.

- MAX_DISPATCHERS

  Specifies the maximum number of dispatcher processes that can run concurrently. Although this parameter exists and some documents do mention it, it actually can be ignored for now.[1] Use the aforementioned DISPATCHERS parameter for now.

- CIRCUITS

  Specifies the total number of virtual circuits that are available for inbound and outbound network sessions.

To identify if there is a shared server performance issue, check the following V$ views:

- V$SHARED_SERVER

  Contains information about shared server processes.

- V$SHARED_SERVER_MONITOR

  Contains statistics information about shared server processes.

- V$QUEUE

  Contains statistics of request queue activity for shared servers.

[1] According to the official Oracle 10g Release 2 database document.

These views provide information about the configuration of a shared server and its current statistics. You can perform some common operations to collect the information:

- Check how many shared servers are currently running.

  Run the following SQL statement:

```
SELECT COUNT(*) "Shared Server Processes"
    FROM V$SHARED_SERVER
    WHERE STATUS != 'QUIT';
```

- The average wait time for a request in the queue.

  Run the following SQL statement:

```
SELECT DECODE(TOTALQ, 0, 'No Requests',
                         WAIT/TOTALQ || ' HUNDREDTHS OF SECONDS')
       "AVERAGE WAIT TIME PER REQUESTS" FROM V$QUEUE
    WHERE TYPE = 'COMMON';
```

If the request wait time is too high, then consider raising the number of shared server processes, that is, modify the SHARED_SERVERS initialization parameter.

Figure 10-6 shows the shared server number and the transactions' processing rate. There are a total of 100 database users connected; as the number of shared server increases, the request can be handled more quickly, because the wait time for a request is much shorter.

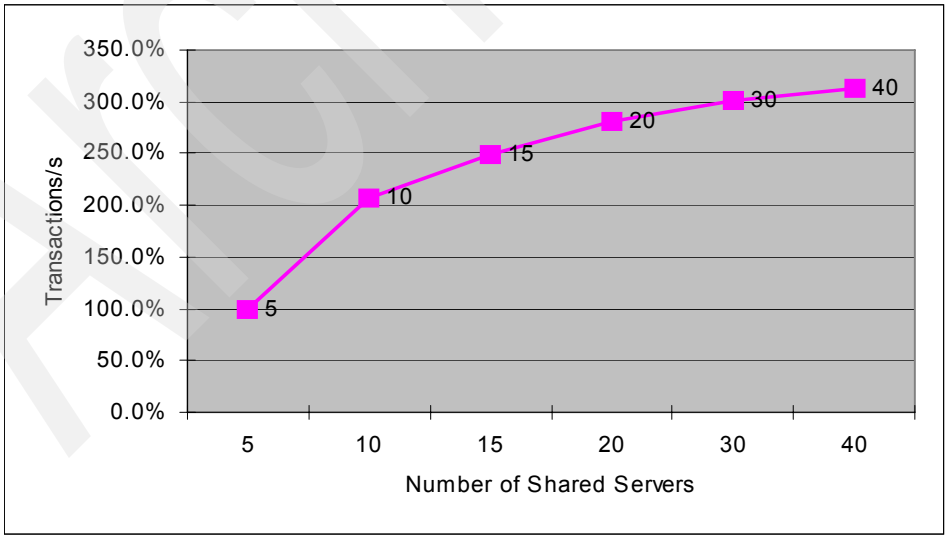


Figure 10-6 Shared servers and transaction processing rate

For more information, please refer to:

Chapter 4, “Managing Oracle Database Processes”, in *Oracle Database Administrator's Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14231/manproc.htm#i1010000

Section 4.3, “Performance Considerations for Shared Servers”, in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/build_db.htm#i22941

Dispatcher configuration

For shared server mode, the incoming connect requests are no longer accepted by the server processes, as in dedicated server mode. Instead, a dispatcher process will get the request and direct it to a pool of shared server processes. An idle shared server process from the shared pool will pick up the request.

Figure 10-7 shows the logical processes of the shared server.

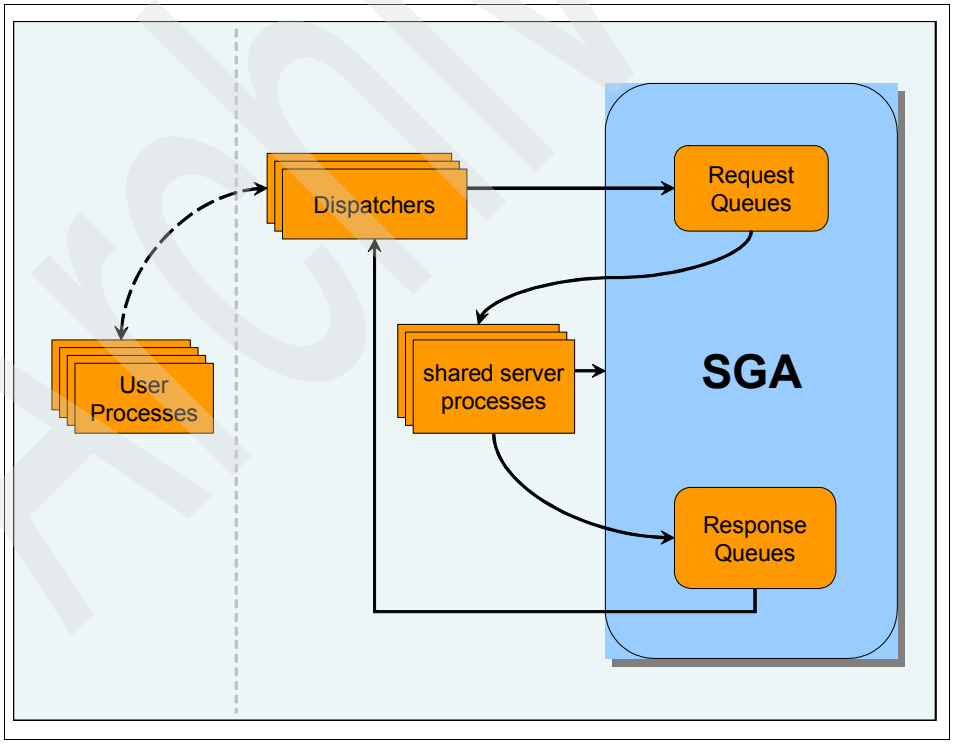


Figure 10-7 The shared server configuration and processes

So in a shared server configuration, how quickly the dispatcher process can handle the incoming connect requests and the server process can pick the requests from the pool are performance crucial. If the server has many client connections being created and destroyed, then make sure you have enough dispatchers to handle them.

To identify a dispatcher related issue, the following V$ views provide the dispatcher performance informations:

- V$DISPATCHER: General information about dispatcher processes.
- V$DISPATCHER_CONFIG: Information about the dispatcher configurations.
- V$DISPATCHER_RATE: Dispatcher processing statistics.

The V$DISPATCHER_RATE view contains current, average, and maximum dispatcher statistics. You may be in one of the following situations:

- If your database provides adequate response time for connections and the current values from this view are near the average and less than the maximum, then you likely have a optimal dispatcher configuration.
- If the current value is close to the maximum, then you might need to add more dispatchers.
- If the current and average values are much less than maximum, then you may have more dispatcher processes than you actually need, so you might consider reducing the number of dispatchers.

There are several parameters that you can tune:

- Total number of dispatcher processes
- Dispatcher connection pooling
- Dispatcher session multiplexing

For example, the following initialization parameter will allow five dispatcher processes to run for the TCP protocol, with connection pooling and session multiplexing enabled, and the limits of each dispatcher is set to 4000 sessions and 950 network connections. The value of TICKS defines the network timeout unit:

```
DISPATCHERS="(PROTOCOL=tcp)(DISPATCHERS=5)(POOL=on)(TICKS=1)
(CONNECTIONS=950)(SESSIONS=4000)(MULTIPLEX=on)"
```

To modify the DISPATCHERS parameter, use the ALTER SYSTEM clause:

```
SQL> alter system set dispatchers=”(protocol=tcp)”;
```

For more details, refer to Chapter 12, “Configuring Dispatchers”, in *Oracle Database Net Services Administrator's Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/network.102/b14212/dispatcher.htm#NETAG012

10.3.4 Optimize memory subsystem

Oracle instance memory tuning is one of the areas where small parameter changes can produce a big increase, as well as a big decrease, of performance. In this section, we describe shared pool, redo log buffer, and database buffer tuning.

Large memory page

The POWER5 processor supports page sizes of 4 KB and 16 MB, and the POWER5+ processor added two new page sizes: 64 KB and 16 GB. Linux system on POWER architecture use 4 KB page size for the operating system and normal applications. For large applications like a database, a great deal of memory needs to be allocated and accessed, which ranges from several hundred gigabytes and to terabytes, so a 4 KB page size is absolutely too small and will make the memory operation inefficient.

Linux has provided a mechanism to enable a database to use large pages (sometimes called huge pages) on SUSE Linux Enterprise Server 9 and 10, or Red Hat Enterprise Linux 4. Example 10-2 shows how to acquire large page information; in the example, the system supports a large page size of 16 MB, and currently no large page memory is allocated.

Example 10-2 Get large page information

```
# grep -i hugepages /proc/meminfo
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:    16384 kB
```

Set the value of the vm.nr_hugepages kernel parameter to specify the number of large pages that you want to reserve. You must specify a sufficient number of large pages to hold the entire SGA for the database instance.

For example, if the SGA size is 2 GB, and the large page size supported is 16 MB, then you should allocate at least 128 large pages to hold the SGA; rounding up to 130 is generally good. Use following command:

```
sysctl -w vm.nr_hugepages=130
```

Whether all the request large pages have been successfully allocated depends on how much continual memory is currently available in the system, so sometimes the system can only allocate less pages than you requested. You need to use this command to check the allocated size:

```
grep -i hugepages /proc/meminfo
```

The HugePages_Total line represents the large pages currently been allocated. If you decide to let Oracle use large page, then modify /etc/sysctl.conf to make sure the system allocates large pages during startup.

Things you need to do to make Oracle use large pages:

- Allocate large pages in Linux, as we described above. Make sure its large enough to hold Oracle SGA.
- Check the kernel.shmall and kernel.shmmax parameters; they should be large enough to hold the large pages that are going to be used by Oracle as SGA.
- Check and make sure the memory lock limit of the Oracle user is large enough; since large pages cannot be paged out to disk, they are virtually locked pages. The memory lock ulimit settings are in /etc/security/limit.conf.
- Start up the Oracle instance. There is no need to modify any Oracle parameters; it will make use of large page memory to allocate SGA.

Figure 10-8 shows the performance gain after enabling large pages. This test was done on a partition with three processor cores and 30 GB of memory; the database size is 10 GB. As the size of the database and physical memory grows, we believe the performance gain will be more notable.

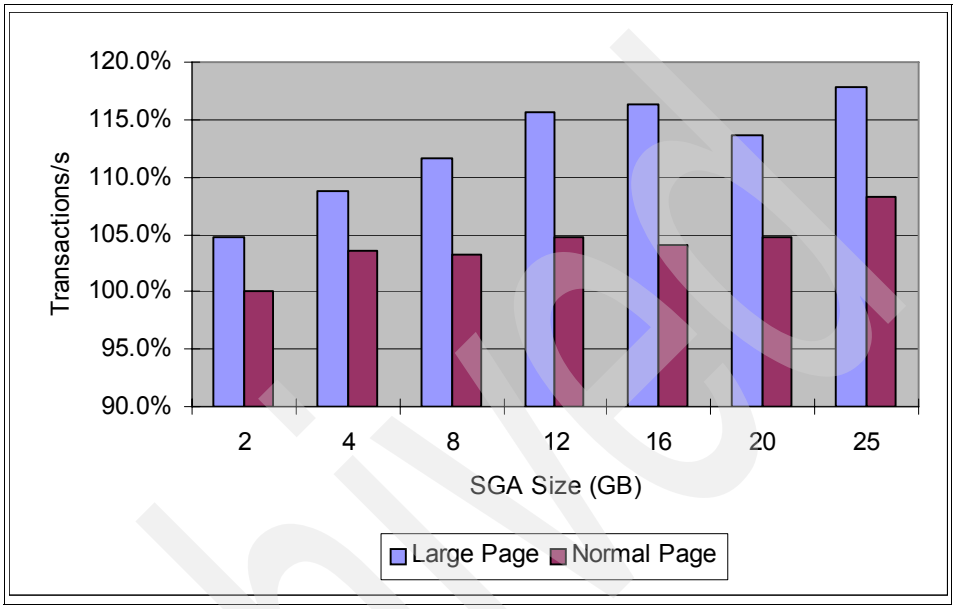


Figure 10-8 Performance gain of large pages

SGA and PGA management

Oracle recommends automatic memory management for SGA and PGA; you just need to specify the size of SGA or PGA, and then Oracle will automatically determine the size for the memory pools and buffer cache areas in SGA and the work areas in PGA.

To use the automatic memory management, set the following two parameters:

- SGA_TARGET

  This parameter specifies the total size of the SGA, and makes the following area of SGA automatically sized:

  - Buffer cache (DB_CACHE_SIZE)
  - Shared pool (SHARED_POOL_SIZE)
  - Large pool (LARGE_POOL_SIZE)
  - Java pool (JAVA_POOL_SIZE)
  - Streams pool (STREAMS_POOL_SIZE)

The value can be memory capacity in units of K/M/G, or set to zero to disable automatic memory management. It can be dynamically modified, but cannot exceed the value of SGA_MAX_SIZE parameter, which is not a dynamic parameter.

- PGA_AGGREGATE_TARGET

  This parameter specifies the total size of PGA; setting it will make Oracle manage the PGA automatically. The value can be memory capacity in units of K/M/G, or zero to disable the automatic management.

Make sure to set the SGA and PGA sizes properly. Too small a value will make the database have not enough memory for data processing, while too big a value will exhaust the physical memory and force the system to swap, which is a huge performance hit.

For more information, refer to Chapter 7, “Memory Configuration and Use”, in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/memory.htm#i49320

Attention: The SGA is a shared memory area, which means it is limited by the operating system shared memory capacity upper limit, that is, the kernel.shmmax parameter we mentioned in “Kernel parameter” on page 320. If you set the SGA size bigger than shmmax, then Oracle may not be able to allocate the SGA and thus fail to start up.

Shared pool

The shared pool is composed of two areas: the library cache and the dictionary cache.

- Library cache

  The library cache is composed of three memory areas:

  – Shared SQL area: Contains the execution plans of parsed SQL statements.

  – PL/SQL programs: Contains PL/SQL programs in compiled forms.

  – Locks.

  Oracle dynamically tunes the relative size of these areas. The only manually tunable parameter is the shared pool global size variable SHARED_POOL_SIZE. To see if the library cache is properly sized, the following simple SQL instruction can be used:

```
select round((sum(pins-reloads)/sum(pins))*100,2) as hit_ratio from
v$librarycache;
```

The term *pins* refers to the number of times a parsed SQL statement was looked for in the library cache. The *reloads* are the number of times the search was unsuccessful. The library cache hit ratio should be as high as 99%. If the hit ratio is lower, either the instance has been recently started, so the cache is suboptimal, or the shared pool is insufficient, and the size should be made larger.

- Dictionary cache

  The following simple SQL instruction shows if the size of the dictionary cache is optimal:

  ```
  select round((sum(gets-getmisses)/sum(gets))*100,2) as hit_ratio
  from v$rowcache;
  ```

  The term *gets* refers to the number of times a request was made for information in the dictionary cache, while the term *getmisses* refers to the number of unsuccessful requests. The dictionary cache hit ratio should be as high as 99%. If the hit ratio is lower, either the instance has been recently started, so the cache is suboptimal, or the shared pool is insufficient, and the size should be made larger.

The statspack and AWR report provide useful statistics information about the library and dictionary caches.

Database buffer cache

The server foreground processes read from data files into the database buffer cache, so the next readings need no I/O operation. Server processes also write modified data into the database buffer cache. Asynchronously, a dedicated background process (DBW*n*) will move dirty data from the cache to the data files. This way I/O performance is greatly increased. The performance benefits obviously depend on cache hits, that is, on how many times server processes looking for data find them in the cache. This section describes the internal structure of the buffer cache and how to tune it.

Buffer cache architecture

The buffer cache is composed of as many buffers as the value of the init<sid>.ora parameter DB_BLOCK_BUFFERS. The size of the buffers are identical and correspond to the init<sid>.ora parameter DB_BLOCK_SIZE. The buffer cache is filled in by foreground processes reading data from data files and flushed out by the DBW*n* process when one of the following events happen:

- DBW*n* time-out (each three seconds)
- Checkpoint
- No free buffer

Data is removed from the buffer cache according to a least recently used algorithm. Moreover, in order to avoid cache quality worsening due to single full table scan instructions, Table Access Full operations are always put at the end of LRU lists.

Optimal buffer cache

In order to see if the size of the buffer cache is optimal, the following query can be used:

```
select name, value
from v$sysstat
where name in ('db block gets', 'consistent gets', 'physical reads');
```

Given the output of this select command, the buffer cache hit-ratio can be obtained with the following simple calculation:

$$\frac{dbblockgets + consistentgets - physicalreads}{(dbblockgets + consistentgets)} \times 100$$

Enlarging the buffer cache

The buffer cache hit-ratio should be 90% or higher. Values between 70% and 90% are acceptable in case it is necessary to resize the buffer cache to improve the library or dictionary hit cache ratios. If the buffer cache hit-ratio is too low, the optimal number of buffers to add to the cache can be obtained with the following query on the V$DB_CACHE_ADVICE view:

```
column size_for_estimate format 999,999,999,999 heading 'Cache Size
(m)'
column buffers_for_estimate format 999,999,999 heading 'Buffers'
column estd_physical_read_factor format 999.90 heading 'Estd Phys|Read
Factor'
column estd_physical_reads format 999,999,999 heading 'Estd Phys|
Reads'

SELECT size_for_estimate, buffers_for_estimate,
estd_physical_read_factor, estd_physical_reads
    FROM V$DB_CACHE_ADVICE
  WHERE name = 'DEFAULT'
    AND block_size = (SELECT value FROM V$PARAMETER WHERE name =
'db_block_size')
    AND advice_status = 'ON';
```

Before running the query, the V$DB_CACHE_ADVICE view has to be activated by means of the following command:

```
SQL> alter system set db_cache_advice=on;
```

Moreover, in order to obtain significant results, a representative workload should have been running on the system for a reasonable time interval, thereby achieving a stable buffer population.

Since the activation of the V$DB_CACHE_ADVICE view has a (minor) impact on CPU load and memory allocation, at the end of the analysis, it is recommended to de-activate the view with the following command:

```
SQL> alter system set db_cache_advice=off;
```

The output of the query is a set of lines showing the incremental benefits of the various cache sizes. The first column of the query output contains the various cache sizes while the latter column shows the physical reads. Upon increasing the cache size, the physical reads decrease, but the incremental benefit also decreases.

Example 10-3 is taken from a small demo system.

Example 10-3 Buffer cache results

Cache Size (m)	Buffers	Read Factor	Estd PhysReads
3	786	1.59	8
6	1,572	1.00	5
9	2,358	1.00	5
12	3,144	1.00	5
15	3,930	1.00	5
18	4,716	1.00	5
21	5,502	1.00	5
25	6,288	1.00	5
28	7,074	1.00	5
31	7,860	1.00	5

Multiple buffer pools

Starting with Oracle 8, multiple buffer pools can be created and separately sized. The database buffer cache is composed of the following three buffer pools:

- Keep pool
- Recycle pool
- Default pool

The keep pool stores data that must not be moved out of the buffer cache. The recycle pool is for data that must be quickly moved out of the buffer cache when it is no longer needed. Everything else is in the default pool. Unlike the shared pool, whose internal memory areas (library cache and dictionary cache) cannot

be separately sized, it is possible to size the keep pool and the recycle pool, and also, as a result, the default pool.

The following example shows how to size a 1000-buffers buffer cache so that 50% is used for the recycle pool, 25% for the keep pool, and 25% for the default pool:

```
DB_BLOCK_BUFFERS=1000
DB_BLOCK_LRU_LATCHES=20
BUFFER_POOL_RECYCLE=(buffers:500, lru_latches:10)
BUFFER_POOL_KEEP=(buffers:250, lru_latches:5)
```

Latches are memory locks and should be sized according to the following rule: 1 latch for each 50 buffers, as in the example above.

Redo log buffer cache

Each server process that updates data first needs to update the redo log files. To improve performance, server processes only write redo log entries into the redo log buffer cache, while the LGWR process is responsible for moving dirty buffers from memory to disks. To avoid buffer data corruption, a locking mechanism (latch) is used, so that only one process at a time can write on the redo log buffer cache. Given the sequential nature of redo log data, only one redo log allocation latch is made available to Oracle server processes. As a result, redo log buffers can be a source of delay, due to high resource contention.

In order to see if there is an excessive redo log buffer contention, the following query can be used:

```
select name, value
from V$sysstat
where name='redo buffer allocation retries';.
```

Any value different from zero shows that processes had to wait for space in the redo log buffer cache.

The size of the redo log buffer can be configured by changing the LOG_BUFFER parameter in the init<sid>.ora file. This parameter gives the value in bytes of the cache and must be a multiple of DB_BLOCK_SIZE.

Each server process wanting to write to the redo log buffer cache must first get the redo allocation latch. The process will then write as many bytes as allowed by the LOG_SMALL_ENTRY_MAX_SIZE parameter in init<sid>.ora. When this number of bytes has been written, the process must release the latch in order to allow other processes to have a chance of acquiring the lock. To increase the ability of server processes to work concurrently, it is recommended that you size the LOG_SMALL_ENTRY_MAX_SIZE parameter as small as possible.

10.3.5 Optimize disk I/O

Here we discuss the optimization of disk I/O.

Hardware RAID level

The IBM System p servers support RAID 0, 5, and 10 for internal disk drives, and the external storage arrays usually support RAID 0, 3, 5, and 10. For database workloads, we usually have two options: RAID 5 or RAID 10. Let us discuss the characteristic of those two:

- RAID 10

  RAID 10 provides good read and write performance, but it has a shortcoming in that only half of the disks carry the data; the other half is the mirror, so the cost is higher than RAID 5. RAID 10 also provides higher redundancy than RAID 5.

- RAID 5

  RAID 5 has lower write performance compared to RAID 10, since each write operation comes with extra read operations and checksum calculation. However, if you are using large external arrays with a smart controller and a large write cache, then the performance load will be quite small. An integrated RAID controller usually does not have these large gears, so RAID 5 is considered to have a lower performance than RAID 10.

Different types of database workloads have different I/O patterns, as we discussed in 10.2.1, “Database workloads” on page 306. In a data warehouse, most disk I/O operations are read, so RAID 5 is a reasonable choice. On the other hand, for transaction processing, there are many write operations, so if you need the highest performance and can stand the cost, then RAID 10 is the best choice.

Oracle Database block size

The Oracle DB administrator is often asked to set the DB block size (DB_BLOCK_SIZE), the one parameter most affecting I/O performance, when in fact, their experience with I/O access patterns may be minimal; the only way to change this parameter afterwards is to create a new DB and move data to it. At installation time, little, if any, information is available on typical I/O size, and the DB administrator may leverage the following basic rule of thumbs (see *Configuring Oracle Server for VLDB* by Millsap):

- The typical DB block size for OLTP systems is 8 KB or 16 KB.
- The typical DB block size for DW systems is 16 KB or 32 KB; sometimes even 64 KB may be recommended.

- A 4 KB DB block size may improve performance only in case of VLDBs consisting of thousands of very small segments.

Moreover, to avoid unnecessary I/O operations, the block size should be a multiple of the operating system basic block size (allocation unit).

As of release 9*i*, Oracle supports multiple block size in the same DB. This capability improves I/O performance, because it is possible to select the best block size for each table space in a database.

Block I/O scheduler

The Linux kernel V2.6 has several block I/O schedulers that have different characteristics, and fit into different workloads:

- Anticipatory
- Completely Fair Queuing
- Deadline
- NOOP

To select a scheduler, use the line:

```
elevator=as | cfq | deadline | noop
```

as a kernel boot parameter.

In general, the CFQ scheduler will give adequate results for most workloads, so it is selected as the default I/O scheduler for Red Hat Enterprise Linux 4 and SUSE Linux Enterprise Server 9 or 10.

8.4.1, “Choosing a file system and an I/O scheduler” on page 220 and 8.4.2, “Tuning the scheduler” on page 231 discuss the affect of I/O schedulers on file serving; the result also applies to the I/O aspect of database workload.

Red Hat has an excellent article, *Choosing an I/O Scheduler for Red Hat Enterprise Linux 4 and the 2.6 Kernel*, that discusses the I/O scheduler and its performance in a Oracle OLTP workload. The article can be found at:

http://www.redhat.com/magazine/008jun05/features/schedulers/

Direct I/O and Asynchronous I/O

Whenever possible, you should enable the Direct I/O and Asynchronous I/O support. These two technologies can help Oracle processes perform disk I/O more efficiently.

Direct I/O (DIO)

When Direct I/O is enabled, file read and write operations will send the data from the application memory space directly to the storage device, bypassing the normal file system buffer cache. This feature is useful for Oracle Database, since it has its own disk buffer cache management mechanism; having both Oracle and Linux maintain a cache is inefficient under a heavy load and a waste of memory.

To enable Direct I/O for Oracle, set the FILESYSTEMIO_OPTIONS initialization parameter to DIRECTIO.

Asynchronous I/O (AIO)

Asynchronous I/O allows the application to submit one or more I/O operations without waiting for it to complete like normal synchronous I/O, and return immediately to execute other tasks. When the I/O operations complete, the application gets a notification and can deal with the result. For Oracle Database, this feature enables more concurrent I/O to be carried out more efficiently.

To enable asynchronous I/O for Oracle Database, set the FILESYSTEMIO_OPTIONS initialization parameter to ASYNCH.

To enable both Direct I/O and Asynchronous I/O, set FILESYSTEMIO_OPTIONS to SETALL.

Figure 10-9 shows the difference between turning AIO+DIO on and off. In this test, the database file is on a ext3 file system. By turning on asynchronous I/O and direct I/O, we can see the transaction processing speed has gone up by 7%, and the I/O wait time has dramatically dropped.

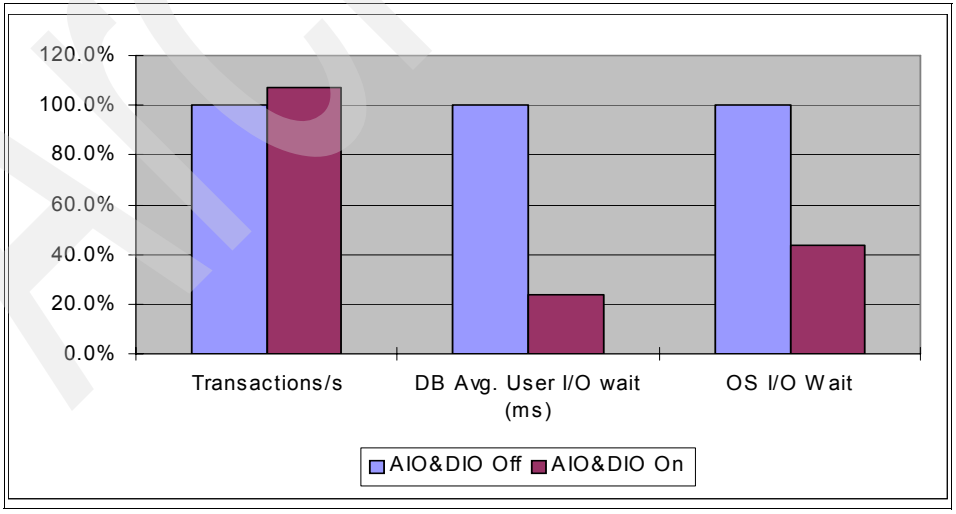


Figure 10-9 AIO and DIO effect

Red Hat has an article, *Tuning Oracle Database 10g for ext3 file systems*, that discusses the DIO and AIO and their performance in a Oracle OLTP workload. The article can be found at:

https://www.redhat.com/magazine/013nov05/features/oracle/

File system versus Raw device versus ASM

There are several ways for Oracle Database to store the data on disks: files on file system, RAW device, and Oracle Automatic Storage Management (ASM).

Table 10-1 shows the difference of the three storage options from the DBA point of view.

Table 10-1 Comparing the storage options of Oracle Database

Options	File system	Raw I/O	ASM
Performance	Normal	Best	Good
Setup	Easy	Moderate	Complex
Management	Simple	Complex	Moderate
Cluster ready	Not all[a]	Yes	Yes

a. Only a Oracle certified cluster or network file system are supported, like NFS and IBM GPFS™.

Figure 10-10 shows the I/O path comparing the file systems raw I/O and ASM.

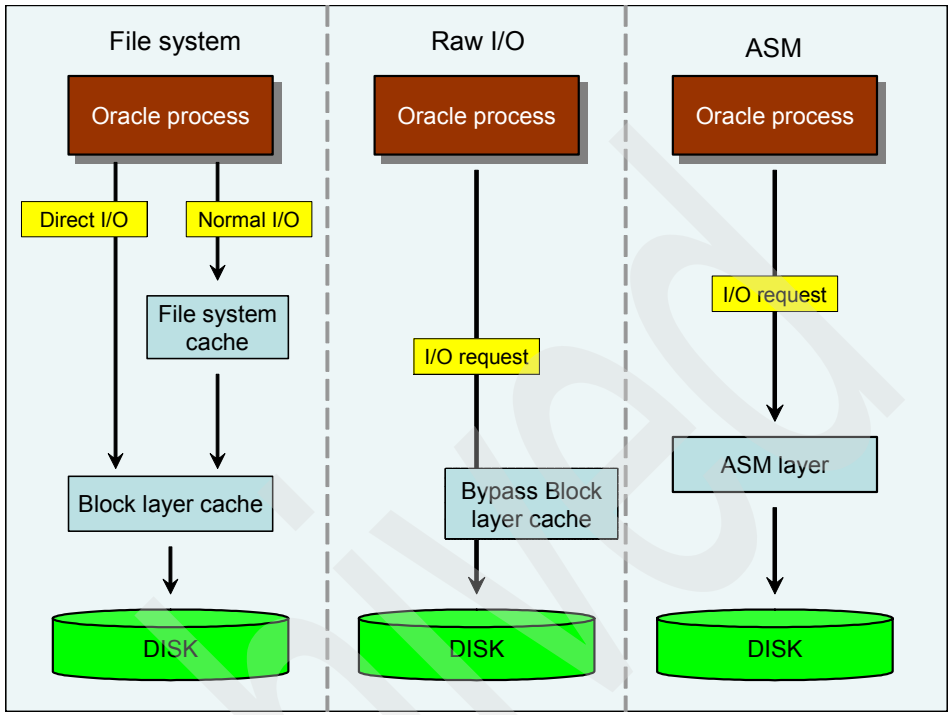


Figure 10-10 Comparing the I/O path of the file systems Raw I/O and ASM

- File system storage

  Using files on an ordinary file system is a easy and straightforward choice for Oracle data storage; any Linux file system that is supported by the distribution can be used. Network File System (NFS) is also supported, so you can run Oracle over NAS device.

- Raw I/O

  Raw I/O means bypassing the operating system caching mechanism; Oracle will read/write the data directly between its memory space and the disk.

  The old way to use raw I/O in Linux is to create a raw character device that is associated with a block device, for example, bind /dev/sdb to /dev/raw/raw1. Then the two devices are just referred to the same physical device, but using different I/O routing.

  The new and preferred way of using raw I/O is just directly use the block device; there is no more need to set up the raw character device. When creating the database/table space, just specify the block device file name as the datafile name.

- ASM

  ASM is a kind of volume manager and file system that is integrated into Oracle, and built and optimized especially for Oracle data files. ASM storage enables near to raw device performance and the easy management of a file system. ASM also provides redundant protection, automatic file distribution, and dynamic configuration that makes it more flexible than ordinary file systems, and the raw device.

  ASM is implemented as a special kind of Oracle instance, that is dedicated for storage management and provides storage access for all other instances that need ASM service. An ASM instance has its own memory structure like SGA and PGA, and background processes. Of course, ASM does not need to handle the request from user processes, so the SGA and number of processes are much smaller than normal database instance.

Figure 10-11 shows the performance difference for a database to use file system storage, raw devices, and ASM managed disks. As we expected, raw device has the best performance, followed by ASM.

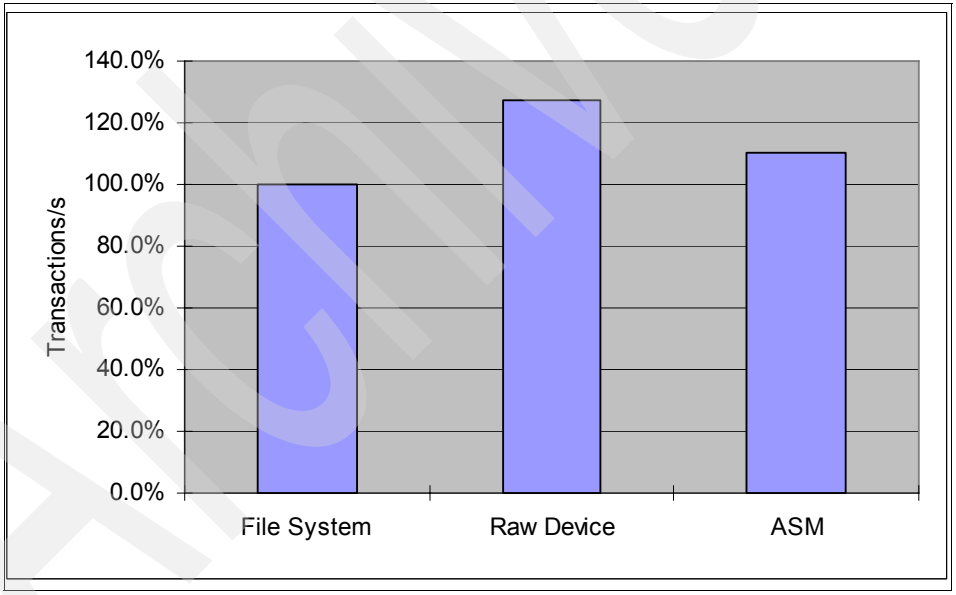


Figure 10-11 Performance difference of file system, raw device, and ASM

Table space distribution

As for table spaces, the basic guidelines in table space design to achieve a good trade-off between the opposite requirements of performance and reliability are:

- Spread data into table spaces so that each table space segment contains data with, as much as possible, similar characteristics:
  - I/O concurrency
  - I/O size
  - Access pattern (read versus write, sequential versus random)
  - Time of life span

  This allows you to easily select the best stripe size unit, and to get the most of the I/O storage subsystem and minimize fragmentation.
- Distribute data in table spaces, keeping in mind table space maintenance requirements, particularly activities requiring the table spaces to be taken offline. For example, since the system table space cannot be taken offline, it is advisable to avoid putting data in it as much as possible.

Given these guidelines, the following is the list of the most important recommendations. See *Configuring Oracle Server for VLDB* by Millsap for further information.

- Put only dictionary segments and system rollback segments in the system table space. This is both for performance and maintenance reasons.
- Put temporary segments in dedicated table spaces.
- Put rollback segments in dedicated table spaces.
- Do not put segments with short lifetimes, causing high fragmentation, in table spaces that are also hosting long lifetime segments.
- Put read-only segments in dedicated table spaces.
- Put similar size segments in dedicated table spaces.

The following query from *Oracle 8 Certified Professional. DBA Certification Exam Guide*, by Couchman may be used to verify which table space is a hot spot (frequently accessed for either reading or writing):

```
select d.name, a.phyrds, a.phywrts, a.avgiotim
from v$datafile d, v$filestat a
where a.file# = d.file#;
```

10.4 Further information

In this chapter, we discussed the means of tuning Oracle Database. However, we only introduced a very limited set of Oracle Database tuning tools; as we mentioned at the beginning of 10.3, “Tuning Oracle Database” on page 318, there are still application and SQL code levels of tuning we can do. And Oracle has a cluster offering called Oracle Real Application Cluster that brings more database power, and of course, more complexity of tuning. In this section, we will have a brief discussion.

10.4.1 SQL tuning

In this chapter, we are mainly talking about the tuning of the Oracle instance and Linux operating system, which will help you access the potential of the system. However, system and instance tuning is only a small area of the whole Oracle tuning. Some user applications may run inefficient SQL code or stored procedures, or have an improper data model and so on. System and instance tuning can barely improve the performance in this kind of situation. Fortunately, Oracle has many tools and documents for SQL tuning tasks.

For more information about SQL tuning, refer to Part 4, “Optimizing SQL Statements”, in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/part4.htm#sthref1059

10.4.2 Oracle Real Application Cluster

Oracle Real Application Cluster (RAC) is the cluster offering of Oracle Database. It allows multiple Oracle instances to access a single database, and provides high availability and scalability. The Oracle RAC is a shared-everything architecture, which means database storage and the in memory data cache are all shared across the cluster, accessible to all the instances.

Figure 10-12 shows the architecture of Oracle RAC.

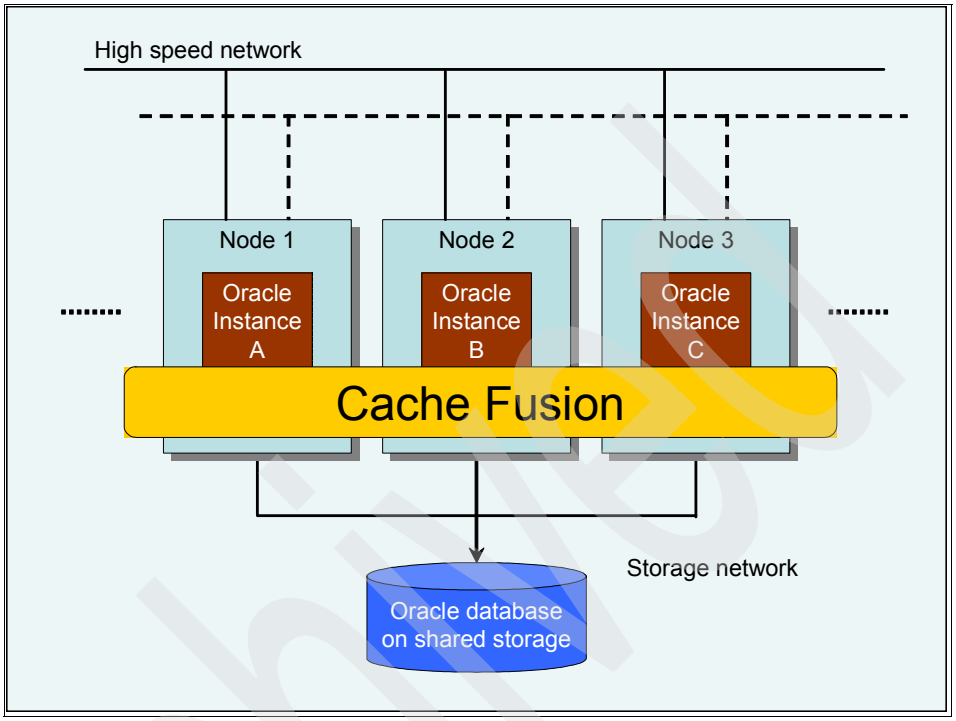


Figure 10-12 Oracle Real Application Cluster architecture

In a Oracle RAC, the following database components are shared among all instances:

- Application table space and datafiles
- System, sysaux, user, and temp table space
- Control files
- Server parameter file

The following components are instance specific and not shared:

- Online redo log
- Undo table space
- Initialization file

Since all the nodes/instances in the cluster access the same database, the Oracle software must guarantee that if one instance modified data and cached the data in memory before writing to the disk, other instances can access the modified version instead of the on disk outdated version. One way is to force the

instance that modified the data to flush the cache to disk before other instances read the data from disk; apparently, it is very inefficient. Another way is sharing the cache among all instances; in Oracle RAC, this technology is called “Cache Fusion”, and it enables all the cluster nodes to fuse their physically separated in memory data caches into a global cache that is accessible for all instances in a coherent way.

The Cache Fusion technology raises the requirement of network connectivity. Since the caches on one node will be accessed by other nodes frequently, the network should be of high throughput and low latency. Gigabyte Ethernet is the minimum requirement, and if possible, a more advanced network interconnect should be used, like 10 Gb Ethernet or InfiniBand.

Please refer to 7.8, “Tuning the network subsystem” on page 200 for Linux system network tuning.

11

DB2 9 data server

To understand the factors affecting the performance of a DB2 database, we must first consider the range of possibilities for the kinds of database servers that may be required. The possibilities reflect the different kinds of operating system environments that are needed by the different applications that can use the databases that the DB2 server needs to provide.

Irrespective of the scale and reach of an enterprise, the primary purpose of any database server will be to store and retrieve data from a storage medium and to allow data to be added to, amended, and searched. As a component of an On Demand Business, the DB2 server may need to provide a high level of concurrent access while maintaining appropriate response times.

Because DB2 9 data server is at the cornerstone of many business environments, it offers comprehensive abilities for integrating business data and for providing a secure and resilient information management system for the organization's information assets. A data server provides software services for the secure and efficient management of structured information. From the applications software perspective, it provides full application development support to programmers working in Java, .NET, C, C++, Cobol, PHP, Ruby, and Perl, in addition to SQL, SQL/XML, and XQuery.

This chapter presents factors affecting the performance of the Version 9.1 release of DB2 database servers running on Linux on System p.

11.1 DB2 9 data server features

There are three editions of IBM DB2 9 data server for Linux, UNIX, and Windows. These are offered to run on System p, System i, or System z servers. The difference between the products lies in the scale of operation supported.

- DB2 Express 9
  - DB2 Express is a full-function relational and XML data server at the entry level.
  - It is scalable on servers, running Linux (or Windows), up to two processors, and can take advantage of up to 4 GB of main memory.
- DB2 Workgroup 9
  - DB2 Workgroup is a full-function relational and XML database server with the same features as DB2 Express.
  - It is scalable to servers running Linux (or UNIX or Windows), up to four processors, and a maximum of 16 GB of main memory.
- DB2 Enterprise 9
  - DB2 Enterprise is a relational and XML database server product with additional special features for high user loads and high availability.
  - The extra features include High Availability Disaster Recovery, the ability to partition tables across different physical servers or logical partitions, and enhanced relational data model with multidimensional clustering, full intra-query parallelism, materialized query tables, and a connection concentrator.
  - It is scalable through the complete range of servers running Linux (or UNIX or Windows) without restriction.

While the three DB2 data server offerings listed include capabilities that serve the needs of most deployments, there are additional capabilities required for certain application types, workloads, or environments that are not needed by every deployment. IBM DB2 9 data server has these capabilities available as optional extra features.

DB2 Workgroup 9 and DB2 Express 9 have value added options for workload management and for 24 x 7 availability incorporating failover.

Businesses running DB2 Enterprise 9 have the benefit of being able to grow and scale across multiple servers by taking the Database Partition Feature (DPF) option. The option carries large performance benefits for environments that must support large databases, complex queries, or both. By distributing a database over multiple partitions, each with their own copies of data, configuration,

indexes, and logs, it is possible to implement tables across multiple SMP servers. By distributing a table across partitions with their separate allocated resources, DPF allows for multiple applications to write simultaneously to different I/O devices or for queries to be factored up and executed in separate pieces at the same time.

For guidance on how to migrate from an unpartitioned (or single-partitioned) database installation to a multi-partition parallel database system, the reader can find a number of migration guides and overviews by searching for the term “db2 dpf” at IBM developerWorks at the following Web address:

http://www-128.ibm.com/developerworks/search/searchResults.jsp?source=google&searchType=1&searchSite=dW&searchScope=dW&query=db2+dpf

Performance Optimization and pure XML database management are additional options for all editions: DB2 Express 9, DB2 Workgroup 9, and DB2 Enterprise 9.

Performance Optimization is focused on complex queries and Online Analytic Processing workloads.

The pure XML feature is for optimized storage and retrieval of XML structured data. The pure XML feature is a technology introduced with Version 9.1 of the DB2 data server.

11.2 Relational and XML data processing capabilities

DB2 9 is the industry's first hybrid data server for managing data from both relational and pure XML formats, supporting both relational and XML operations.

DB2 has been providing high-performance data storage and access for relational data based on ANSI SQL standards and data storage optimizations, such as data partitioning and advanced indexing and query optimization techniques. DB2 9 data server products feature an optimized storage engine for XML alongside the relational database engine.

XML data is often used for inter-application data exchange and document management purposes. The flexible and self-describing nature of XML data makes them ideal for many application scenarios. The standards for how XML data can be queried are:

- XPath Version 2.0, including the XPath Data Model
- XQuery Version 1.0, which extends XPath

Application developers can store XML data directly inside a DB2 server and reap the benefits of transactions, advanced data resiliency, secure access, and the ability to search large amounts of XML data using XQuery.

DB2 9 data server supports XML open standards and features native XML data store support.

In particular, XML processing in DB2 may make use of:

- The newly introduced XML column datatype, which allows application programs to search for elements within XML parameters. The values held in an XML column retain their hierarchical structure.
- Values in XML columns may be used as arguments to SQL/XML functions and also SQL/XML functions may return XML data results. SQL/XML functions include user defined functions and stored procedures coded in any of:
  - Java (JDBC or SQLJ)
  - Cobol
  - C or C++
  - C# and Visual Basic®
  - PHP
- Management of XML schemas and DTDs into a repository to allow validation and processing of XML documents as values in XML columns.

Database developers may work hybrid SQL/XML technology by using the Version 9.1 DB2 data server, provided:

- The database installation is unpartitioned (or single-partitioned).
- The XML added value option has been taken.

For further information, the reader should consult the *XML Guide and the XQuery Reference* available with the product documentation. These are available for download at:

`http://ibm.com/software/data/db2/udb/support/manualsv9.html`.

11.3 Configuring DB2 on Linux for performance

In the present chapter, we consider performance factors when using DB2 Enterprise 9 on an SMP System p Linux server.

The documentation provided with DB2 9 includes a performance guide, *DB2 Version 9 for Linux, UNIX and Windows Performance Guide*, which reviews performance from the planning stage, to tuning parameters and to query optimization strategies. The performance guide may be obtained with the DB2 data system 9 product deliverables, or downloaded from:

`http://www.ibm.com/software/data/db2/udb/support/manualsv9.html`.

These are the overall stages to achieve performance of a database for application workloads under Linux on System p:

- The planning stage encompasses how the resources are managed by the database system. This involves determining the physical storage, and how it is to be allocated and structured to support the organization of tables for growth and retrieval.
- Query optimization involves decisions about the performance of the important statements in the application, expressed in SQL, XQuery or SQL/XML, and the access plan which they execute.
- Tunable configuration parameters exist to improve the performance of the database once commissioned and maintained in a running environment against the production workloads supported by the database applications. The majority of these may be regulated automatically.

Below the data structuring level of database application performance, there is the question of I/O performance, and system processor and memory tuning on System p SMP boxes. System monitoring tools may reveal the performance characteristics giving rise to performance bottlenecks.

For illustration purposes, we shall focus on the situation of physical servers when there are up to four physical cores available to run an example database with a number of physical disks. We consider the situation where the disks are mounted as distinct logical volumes. Each such logical volume represents a virtualized set of physical disks that have been combined using a RAID 5 array arrangement.

Changes that may have a big impact on database performance may arise from:

- Application design and enablement
- Processor and memory resource allocation
- Disk resource allocation
- I/O design and tuning

Before reviewing how changes to the baseline level of performance may be effective, we should first look at how the database should be configured according to the envisaged application requirements and available hardware.

11.3.1 Overview of performance planning

Appendix A of the *DB2 Version 9 for Linux, UNIX and Windows Performance Guide* provides the complete set of available configuration parameters.

For practical purposes, there is a good likelihood that a great number of database parameter settings may be left entirely under the influence of the database system in the course of defining the database. Unless there has been prior performance experience at a production site that has highlighted specific configuration steps, the standard administrative preference will be to allow DB2 to automate changes to the system.

By default, a DB2 9 database monitors performance-critical measures so as to attempt to anticipate problems before they happen.

The following automatic features are enabled by default:

- Configuration advisor

  This tool is automatically invoked at database creation time to set the database configuration parameters to tailored values instead of out-of-the box defaults. It can set up initial memory allocation parameters that are crucial to ongoing performance. The tool may be invoked again after database population or significant content or application changes.

- Automatic statistics collection

  By default, DB2 maintains up to date table statistics. Depending on the workload, the RUNSTATS utility is automatically invoked in the background on a table prioritized basis. The DB2 SQL query optimizer can then formulate an up-to-date access plan resulting in optimum query performance.

- Automatic storage

  This provides administrative simplification of storage mapping to physical disks, as discussed in 11.3.2, “Overview of storage efficiency” on page 353.

- Self tuning memory

  In Linux systems, this allows available memory to be better distributed internally throughout the individual database system.

- Health monitor

  This is a database server tool that can “call out” significant changes in the database environment between snapshots.

- Utility throttling

  This regulates the resources utilized by maintenance tools, so that they can run at the same time as production databases are executing. Statistics collection, backup operations, and rebalancing may all be throttled.

The automatic features may be individually enabled or disabled. Collectively, these features are switched on provided they are not unset from the default installation of DB2 V9.1 database servers.

11.3.2 Overview of storage efficiency

There are a variety of storage options that IBM provides for the purpose of resourcing a DB2 database server with adequate storage for data. These include:

- Internal SCSI disks
- External hard disk arrays
- Network attached storage options
- High performance fast storage options

Ordinarily the internal SCSI disks are obtained with the purchase of a System p machine and are used for the installation of the base operating system and installation of software packages. For business data processing purposes, non-internal options are recommended for database storage. The scale of the enterprise database will shape the decision of which of these options will be available to the database administrator.

The decisions about storage mapping to physical or virtualized hard disks is a crucial decision at the planning stage.

There are three performance-critical areas where the database analyst has important choices to make: table spaces, logs, and buffer pools.

Table spaces

A table space is a data structure that holds data belonging to a single database. There are a minimum of three table spaces holding the data belonging to one DB2 database.

Each database may use up to five different kinds of table spaces. The table spaces that are most specific to the application data are:

Regular table spaces
: These hold persistent data and indexes.

Long table spaces
: These can be provided to hold long column data or large object (LOB) column data, such as binary media files or scanned documents.

The long table space is not needed by every database depending on the complexity of the columns that it defines. However, using long table spaces where there are long or LOB data values in the database is a performance improvement.

In many installations, it can be a common practice to define many regular table spaces. In particular, a division into one or more regular table spaces for data and a regular table space for indexes is often effective in increasing performance, provided sufficient disk resources exist.

At database creation time, there are two further table spaces:

Catalog table space
: This is a single table space created by the DB2 system at the time the create database command is issued. This storage is for meta-data about the database.

System temporary table space
: This is for holding intermediate table data while sorting or collating results.

An optional extra table space is for user temporary objects:

User temporary table space
: These store declared temporary tables as created by the SQL statements issued by applications.

Table spaces involve allocated storage from the available disk I/O resources. This can be managed either by the operating system or by the DB2 system.

The decision about which scheme of table space management to use may be made per table space. In practice, however, the decision about how to manage table spaces is usually characteristic of the database analysis employed by the database administrator at the installation.

The three schemes of table space management have different administration and performance implications, as follows:

- System managed space (SMS)

  In this scheme, table spaces are managed by the operating system. Table spaces are made up of one or more containers. Each container represents the mapping onto physical storage. With SMS, each container is a file system. The operating system allocates each container's file storage through ordinary system calls. Consequently, the operating system will control how the I/O is buffered, how space on the storage resource is allocated, and how storage space is extended automatically, as the database grows.

- Database managed space (DMS)

  In this scheme, the DB2 system provides the containers that make up each table space. The containers are either files (DMS-file) or the containers are unformatted by the operating system (DMS-raw).

  Containers for DMS-raw should be block devices presenting complete unformatted contiguous storage, as seen in Linux by fdisk.[1]

  Containers for DMS-file are files whose size is set up at the time that the table spaces are first created.

  The allocation of table spaces to a database must occur before the database is populated with rows. Consequently, it is necessary to analyze the size of the file or device needed to hold each of the table spaces.

  If the size turns out to be miscalculated, it is possible to increase or decrease the given table space using the ALTER table space command in DB2.

- Automatic storage

  Where a table space is defined as MANAGED BY AUTOMATIC STORAGE, the DB2 system is responsible for providing storage from one or more storage paths. The DB2 system is then responsible for maintaining an appropriate container size.

As a generalization, SMS is biased towards easier administration and DMS is biased towards greater performance. The response time differences between the different schemes for most anticipated workloads are believed to be several percent in most Linux environments. The automatic storage scheme is recently introduced (with DB2 V8.2) and promises to offer a good balance of performance and ease of administration.

[1] Raw-character is also permitted, although deprecated. See "Changing raw devices to block devices (Linux)" in the DB2 9 Infocenter at:
http://publib.boulder.ibm.cnm/infocenter/db2luw/v9/index.jsp

Using SMS to hold catalog and system temporary table spaces is usually recommended. The selection of the table space storage scheme (SMS, DMS, or automatic) to implement for regular and long table spaces may be decided by the analysis of the data to be stored. Segregating different tables that are potentially high volume request points into different table spaces can benefit performance greatly.

In the case of SMS and of DMS-file, the decision about which file system to employ is a performance factor. With Linux, the ext3 system is a familiar option, although xfs may be utilized, as may reisser or any performing file system.

For managed storage, the transfer from or to the store occurs as pages. A table space page size may be set at 4 KB, 8 KB, 16 KB, or 32 KB. The choice of page size determines whether a row occupies multiple pages, or whether a page contains multiple rows. Storing a row, or many rows, in a single page, allows the entire row or rows to be retrieved as a single unit from disk.

Large page sizes are good for storing tables with very long rows or tables used regularly for sequential data access. However, storing small, randomly accessed rows in large pages makes for inefficient use of the buffer pools. If there is a need to allow different tables to be stored with different page sizes and extent sizes, and to allow separate buffering, it is recommended to use multiple table spaces.

Logs and disk subsystem utilization

Log files are part of the transaction recovery control program of DB2.

Logs record all transactions as they execute against the database. They are essential for transaction control because they need to be consulted every time a transaction rollback occurs. They can be retained to allow roll-forward recovery. In the event of a database failure, the logs are used to return the database to a consistent state.

It is highly recommended that logs are placed in a different physical location from table spaces. Typically for performance, the logs need to be kept in storage, which works at similar speeds for the storage used for table spaces.

The best way to get high performance in disk access is to place the DB2 storage capabilities separately onto dedicated physical disks or attached storage systems. Specifically:

- The operating system and system software, usually on the same disk as the boot partition, for example, /dev/sda, if this is the hard disk to boot from.
- Paging file (swap file or virtual memory), especially if there are other applications and services running on the same machine or LPAR as the database server.

- One or more dedicated separate physical disks as mount points for file systems (or, alternatively, as raw block devices) for table space containers.
- Database inserts and updates are accompanied by log file activity, so separating log files from data makes a big difference, especially in Online Transaction Processing systems.
- Secondary log files should ideally be kept on another hard disk because they are a means of disaster recovery in the event of failure and corruption on the primary log file store.

Any or all of these could be on RAID arrays for improved reliability and performance. Even a small system may have a heavy OLTP load, constituting many data writes, which benefit from greater parallelism through RAID or higher performance storage. External storage usually gives the highest payoffs, especially in the case of table spaces and logs.

11.3.3 Use of memory

Memory is needed to dispatch agents for serving DB2 processor requests. A further use of memory is to hold prefetch pages in order to reduce the number of fetches from physical storage.

Buffer pools

Buffer pools in DB2 provide in memory caching of indexes and most data (excluding LOBs).

When a new row is added to a table in the database, it is first added to a new page in the buffer pool. When rows are retrieved in response to a query, the pages that contain them are held in the buffer pool before passing on to the requester. Every available buffer pool is searched to see if the data needed to satisfy a request is found on existing pages. If there is no hit, then the rows are retrieved from storage to be copied to the buffer pool before being passed to the requester. Long lived data is eventually retired when space in the buffer pool has been filled and needs to be overwritten.

The benefit is that whenever the data is found in pages already held in the buffer pool, they are immediately passed to the application that requested them.

If the buffer pools are adequate, this allows data requests to be filled from memory rather than disk. Buffer pools use fixed page sizes, which must be the same as the page size of any table spaces they serve.

Every DB2 database makes use of at least one buffer pool. A buffer pool belongs to exactly one database and is available to be used by any number of its table spaces.

With DB2 V9.1 for Linux, an existing buffer pool (called IBMDEFAULTBP) of 4 KB pages is already defined as part of the system catalog at database creation time. It is available to associate with any or all table spaces with a page size of 4 KB. Additional buffer pools should be created for the page size of every table space needed for the database.

Buffer pools for a database need to be defined to match the page size(s) of all table spaces.

In DB2 9, table spaces may be defined with any of the page sizes given in Table 11-1. A single page size for the entire database could be appropriate if the row sizes of tables are similar. As an example, if we create a particular database using 16 KB pages, this will be treated as the default for any buffer pools defined for that database.

For the definitions to take effect, the buffer pool sizings need to be completed before the database is activated. However, it is possible to add additional buffer pools later along with any table space that is added.

As regards table space design, multiple page sizes were the recommended approach for earlier releases of the IBM DB2 product. The standard choices are 4 KB and 16 KB buffer pool sizes for online transaction processing (OLTP) kinds of processing. Otherwise, larger page sizes of 8 KB and 32 KB are usually beneficial. However, a single page size, such as a 8 KB or 16 KB page size, is a practical possibility for DB2 V9.1 database servers for many kinds of applications.

Where the implementation needs a smaller scale of database operation, then a 4 KB or 8 KB page size may work better, because the size of each row may not justify holding a larger page. In larger installations, the decision will be influenced by the maximum size to which each table can grow. These maximum permissible sizes are given in Table 11-1.

Table 11-1 Page size for maximum size of table in DB2 Version 9.1

Table space page size	Maximum size of table
4 KB	2048 GB
8 KB	4096 GB
16 KB	8192 GB
32 KB	16384 GB

Provided that we have a good mapping from the table spaces to the buffer pools defined for their pages sizes, increasing the size of their buffer pools is beneficial, especially for OLTP applications.

This is because with a larger size of the buffer pool, the pages that it caches stay in the pool longer before being retired. The greater the number of hits in the buffer pool, the smaller will be the number of I/O operations to fetch database objects into memory. However, there is a tradeoff between increasing the size of a database's buffer pool(s) and the total available memory, including the database heap.

In usual practice, increasing the buffer pool size from the installation default is the most significant single tuning step. Here is an example of a database instance connected to an initial database. The size of the buffer pool(s) can be viewed as seen in Example 11-1.

Example 11-1 Determining buffer pool size

```
db2inst1> db2 "select * from syscat.bufferpools"
BPNAME BUFFERPOOLID DBPGNAME NPAGES PAGESIZE ESTORE NUMBLOCKPAGES
BLOCKSIZE NGNAME
------------ ------------------ ---------------------- ---------
-------- ----- -------
- ------------------------ -------------- ------------
IBMDEFAULTBP 1 -                    4096 N                0
0 -
BP4K
3 -
-2        4096 N                 0          0 -
```

For production purposes, much larger buffer pool sizes are to be expected. To create a new buffer pool, the database administrator needs to use the CREATE BUFFERPOOL command.

There are several performance issues to balance in deciding on the number and size of buffer pools.

The small size of the default buffer pool ensures that starting the database is possible. The administrator needs to exercise caution not to use more memory than is available, in order to avoid causing the database instance to start up next time with reduced resources (SQLSTATE 01626). So all the buffer pools created must total less in size than available memory.

top, nmon, or the graphical tool gnome-system-monitor can all be used to give a quick visual check that memory is not being over-allocated.

If the database design has table spaces with differing page sizes, then each page size requires at least one buffer pool.

If the database applications are known to frequently re-scan a particular table, then a good practice is to keep it on a separate table space with an extra independent buffer pool serving it. This principle applies to any frequently hit “hot spot” tables that benefit from separating into different containers and associated buffer pool memory.

11.4 Database administration for performance

All administration of an individual database is through its instance. Additionally, the instance may be used to alter the configuration of the database manager.

In Linux systems, the default user account for running a instance is db2inst1. The set of instances currently created on a DB2 server is seen by running:

```
/opt/ibm/db2/instance/db2ilist
```

If the DB2 setup is new and we need to create or re-create a first DB2 instance called db2inst1, then we can run:

```
/opt/ibm/db2/instance/db2icrt -u db2fenc1 db2inst1
```

where db2inst1 and db2fenc1 are predefined users in non-administrative groups. On Linux on System p, this will create a 64-bit DB2 instance managed by db2inst1.

Note: A DB2 database instance

A database instance is an independent, lexically closed environment for naming, creating, and accessing database objects, such as databases, tables, and views. An instance behaves as the controller of the individual databases that are named in it, responsible for starting or stopping those databases, and determining the access privileges of users to send commands to its databases.

In DB2, each instance is provided by an operating system defined user with appropriate privileges.

For performance reasons, it may be beneficial to use one instance to create one database, although this becomes less significant with smaller scale databases (for example, below hundreds of GB).

Once the instance has been created, database creation proceeds by using the **create database** command. Full syntax details are available from the *SQL Reference Manuals Volumes 1 and 2*, which a re available with the product documentation or by download at:

http://www-306.ibm.com/software/data/db2/udb/support/manualsv9.html

To see the parameters that are available to the SAMPLE database (provided as an optional example in DB2 installations), the DB2 instance may call the **get database configuration** command, as shown below:

```
db2inst1> db2 get database configuration for sample
```

Additionally any DB2 instance is allowed to view the database management configuration, as shown below:

```
db2inst1> db2 get database management configuration
```

The instance may revise any of the parameters shown by using the UPDATE command.

The user of an instance may wish to review the settings of three groups of variables and parameters:

- The DB2 Profile variables for the current instance

  These control some aspects of configuration, such as the identifiable name of the server system which may be administratively useful. In addition there are settings for switching on TCP/IP which may be critical. Further settings, such as DB2_PARALLEL_IO, may be important to enable particular optimizations.
- The database parameters
- The database manager parameters

The DB2 Profile variable settings may be reviewed by running:

```
db2inst1> db2set -all
```

For most purposes, including JDBC application support, our server will need to be using TCP/IP. We can switch this on, for all instances hosted at the server machine, by running:

```
db2inst1> db2set DB2COMM=TCPIP
```

For administrative clarity, it may be useful to provide default settings for data and diagnostics in the database manager. For example:

- We could set up a default path for creating a database, as follows:

  ```
  db2inst1 > db2 update dbm cfg using DFTDBPATH /Data
  ```

- We could decide where the diagnostic troubleshooting and problem determination logs go, as follows:

  ```
  db2inst1 > db2 update dbm cfg using DIAGPATH /Logs/db2dump
  ```

Considering the impact on performance tuning, our assumptions throughout this chapter are that:

- Automatic settings and automatic tuning is adopted where available.
- Any instance is used to create exactly one database.

Our initial examples will consider an unpartitioned database.

11.4.1 Tuning functioning databases

Even a fully functioning database that is activated and serving requests may encounter resource barriers.

One example of an under-resourced database server would arise if there is no sufficient available memory not provided for buffer pools, prefetching, and database agents to execute queries. Another example would be insufficient disk capacity. Under-resourced databases, provided they can execute, will produce failure conditions and SQL errors that may be looked up to decide remedial action.

In addition, there are some initial settings that are not checked automatically. In particular, the setting of application heap size is 128 x 4 KB pages by default. If larger applications are likely to execute against the database, then the application heap size may need to be increased. For example, against our TRADE6 database, we might issue the commands shown in Example 11-2.

Example 11-2 Increasing application heap size

```
db2inst1 > db2 connect to TRADE6
db2inst1 > db2 update db configuration for TRADE6 using APPLHEAPSZ 512
db2inst1 > db2 connect reset
```

It is not always possible to predict the limits that will be encountered by production runs against a database, so extensive pre-testing of a database, possibly using a separate DB2 instance configured on the final resources, is the expected practice.

Provided that there is not one simple overpowering resource bottleneck, then DB2 V9.1 tuning on Linux needs to make appropriate adjustments to:

1. The DB2 instance's registry variables
2. The database manager configuration parameters
3. The database configuration parameters
4. Allocation of memory to buffer pools
5. Refreshing the accuracy of system catalog statistics

Adjustments 2 to 5 can be made to DB2 databases automatically.

We need only to provide supplementary guidance on the systems affected.

11.4.2 Memory management

As shown in Figure 11-1, memory in DB2 9 is organized in four different memory sets

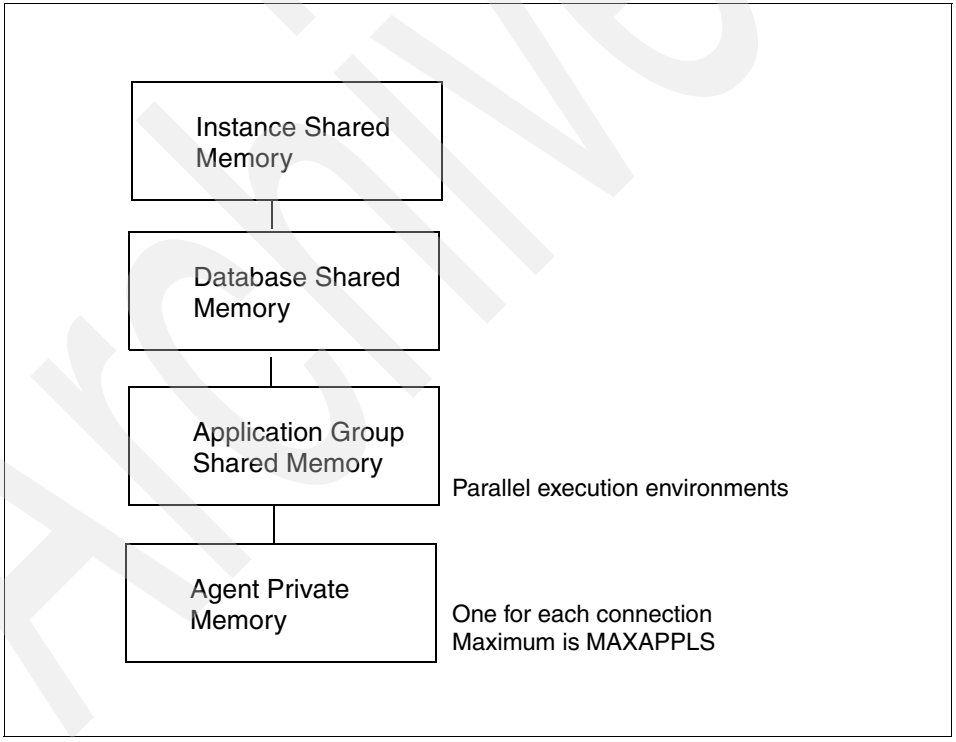


Figure 11-1 Memory organization in DB2

Instance shared memory

This is unique to every instance and allocated at instance startup. It is made up of instance memory, a monitor heap for use by monitoring, and an audit buffer.

Database shared memory

This is unique to the database and allocated when the database is activated. It holds all the buffer pools. It is a space needed for all database specific commands, including SQL execution, and locking and running utilities, such as restoring from media.

Application group shared memory

This only exists in environments involving specific parallel execution within or between partitons. These environments are either DPF enabled or use Connection Concentrator or have set up for parallelism within a single partition. For unpartitioned environments, there are only three memory sets.

Agent private memory

The memory needed by every agent to do work. An agent is created every time there is a new SQL connection. There are as many agents as there are connections, whose maximum number is set by the MAXAPPLS parameter of the database.

The DB2 9 memory set size limits are relatively great based on a 64-bit architecture for Linux on System p. So, it is usually physical memory that imposes limits, because all the memory pools must fit into the available user memory.

Additionally, it is desirable to check that the kernel setting of shmmax allows sufficient available memory to be addressed.

In DB2 on Linux, the self tuning memory management (STMM) feature affects memory allocation within the particular instance. The STMM dynamically distributes available memory resources amongst the instance's memory consumers.

These memory consumers are:

- Sort routines
- The package cache
- The lock list
- Buffer pools

Memory adjustments may occur within the instance as a result of changing workloads, such as a change from OLTP mode to batch mode. Memory utilization within the instance, such as the allocation to buffer pools, may be adjusted by STMM, provided that autoconfiguration is left on.

However, the problem of adjusting memory to coexist with other memory loads, such as those from other applications or other database instances, at the database server machine is not solved by STMM under Linux on System p. If there is a need to balance the tasks of the database server with the memory requirement of other applications, then a satisfactory approach may be to use dynamic LPARs to run each database instance.

It should be remembered that memory reallocation will require a reboot of the affected system images under System p virtualization. So the dynamic LPAR approach is suggested here as a possible working practice and is not intended as a fully automated solution without downtime.

We investigate the dynamic LPAR approach, and look at its trade-offs in 11.7, “Database guidance for performance on LPARs running Linux” on page 374.

11.4.3 I/O Parallelism for RAID disks

Performance can be optimized when table spaces are contained on Redundant Array of Independent Disks (RAID) storage. Ideally, each RAID device should be mapped to a single container (a raw device or a file system) for defining table spaces.

At the entry level, efficient I/O systems will commonly employ a stack of physical disk drives with solutions such as RAID 5 to provide for striping across a set of available disks. In the scenario we consider, this is achieved by the PCI-X controller of several RAIDed disks defined to the VIO Server.

For more highly performing solutions, the reader should consult http://www.ibm.com/storage or *Deployment Guide Series: TotalStorage Productivity Center for Data*, SG24-7140. The general characteristic of all these solutions is that a RAID operation is defined at the physical adapter for the storage medium, in preference to defining it at the Linux operating system level. Therefore, the DB2 instance needs to be told to take advantage of the parallel I/O.

A registry variable should be set by the instance, for example, as follows:

```
db2inst1 > db2set DB2_PARALLEL_IO=*
```

DB2_PARALLEL_IO forces DB2 to use parallel I/O for the RAID disk, since otherwise the striping across disks is ignored. The asterisk wildcard (*) is used where we need to ensure that all table spaces are contained by parallel devices. Otherwise, we may number the table spaces in accordance with their Tablespace IDs returned by the LIST TABLESPACES command.

By default, the assumed number of spindles is six. If we want to change this to five, we can make the following setting:

```
db2inst1 > db2set DB2_PARALLEL_IO=*:5
```

The purpose of providing this setting is to increase the number of consecutive pages fetched into the buffer pool in a single I/O.

There are two crucial table space configuration parameters that now need adjustment.

EXTENTSIZE

The extent size for a table space represents the number of pages of table data that will be written to a container before data will be written to the next container. This needs to be set at RAID stripe size X number of disks in RAID array / Buffer pool page size. So, if the disks are RAIDed at a 16 KB stripe size at the VIO Server, and a buffer pool of 4 KB is provided, then the extent size can be set to equal four times the number of disks.

PREFETCHSIZE

The number of prefetched pages for each table space. This needs to be set at Extent size X number containers used by the table space. By default, the PREFETCHSIZE will be set to AUTOMATIC.

Prefetching may be optimized on Linux to avoid buffering loads usually associated with Linux file systems. We investigate our tuning options for reducing these loads in 11.7.5, “I/O subsystem settings” on page 380 later in this chapter.

The database manager monitors buffer pool usage to ensure that prefetching does not remove pages from the buffer pool if another unit of work needs them. To avoid problems, the database manager can limit the number of prefetched pages to less than we specify for the table space.

The prefetch size can have significant performance implications, particularly for large table scans. The database system monitor and other system monitor tools are available to help tune PREFETCHSIZE for the table spaces.

A suitable number of I/O servers, using Linux asynchronous I/O libraries (as discussed in “Asynchronous I/O” on page 381) for prefetching is based on the number of physical disks. By preference, it is effective practice to leave NUM_IOSERVERS set to AUTOMATIC.

11.4.4 Checking the recommended values with AUTOCONFIGURE

For any DB2 instance, once our database has been created and has had a first run to populate it, we can re-evaluate the settings we made through the AUTOCONFIGURE command. In addition to evaluating the settings, we can apply the changes only to the database, if we so choose, or to the database and the database manager, if we so choose.

At its simplest, we can run:

```
db2inst1 > db2 autoconfigure apply none
```

which returns a list of the current setting of buffer pool size(s) and configuration parameters of the connected database.

Periodically, through load testing of the TRADE6 database, we found it useful to run:

```
db2inst1 > db2 autoconfigure using mem_percent 60 apply db and dbm
```

This applies the recommendations, including statistics and a shared memory allocation, based upon 60% utilization of available RAM, to the database and the database manager.

Repeatedly applying the command to a running database results in a statistics update that is necessary for its continued performance. The statistics are gathered during the execution of each SQL statement by the SQL optimizer to perform a number of important calculations. In particular:

- The size of each table accessed and the column datatypes defined.
- Whether indexes have been defined for the referenced tables.

The objective of the optimizer’s calculation is to infer the most efficient access plan. If the statistical information is out of date, then the access plan will potentially result in unneccesarily long execution times for SQL statements. This becomes extreme in the case of complex queries that may benefit either from applying AUTOCONFIGURE frequently or possibly invoke the **runstats** utility. For example, **RUNSTATS ON mytablename WITH DISTRIBUTION AND INDEXES ALL**, assuming we need to make potential updates to all indexes.

After issuing AUTCONFIGURE (or RUNSTATS) against the TRADE6 database in our example, it was necessary to rebind all packages for the applications that have database variables executing in their SQL operations. This was done as shown in Example 11-3.

Example 11-3 Rebinding all packages

```
db2inst1 > db2 connect to trade6
db2inst1 > db2 bind /opt/ibm/db2/V9.1/bnd/@db2cli.lst blocking all
grant public
db2inst1 > db2 connect reset
```

11.5 Application support to DB2 for Linux on System p

On System p servers, since Version 9.1, all DB2 instances are 64-bit programming architectures.

The provided programmatic interface is 64-bit wide. Therefore:

- Client access connections (as defined by a DB2 client application)
- User defined functions
- Stored procedures

must be built to take advantage of the 64-bit target DB2 server.

User defined functions may be coded in SQL, C / C++, or Java. In the case of stored procedures, these may be SQL, C / C++, or SQLJ.

For Java programmers, a 64-bit Java virtual machine (JVM) is provided with the DB2 Version 9.1 release. A JVM is the execution environment for binary Java bytecodes on which all application performance characteristics are based. 64-bit JVMs have the advantage of allocating larger heaps for object growth, prior to garbage collection, and allowing larger address spaces so as to improve memory limits where disk retrievals might have otherwise been necessary.

In correspondence to the supplied JVM, the Java programming strategy conforms to the Java 2 Platform Standard Edition 5.0 specification. This includes the Java Database Connectivity (JDBC) 3.0 API.

11.5.1 JDBC driver features

DB2 9 data server supplies the IBM DB2 Driver for JDBC and SQLJ. For Linux on System p, this code is a 64-bit implementation of the JDBC API.

The Java 2 Platform Standard Edition (Java SE) 5.0 specification permits four alternative types of JDBC drivers. The four types are:

- The type 1 driver, alternatively described as the JDBC-ODBC bridge driver, at:

  `http://java.sun.com/j2se/1.5.0/docs/guide/jdbc/bridge.html`

  is supplied with the Java SE 5 reference implementation for experimental purposes. Other types of drivers are more closely associated with vendors of data systems.
- The type 2 driver, alternatively described as the Native API driver. This driver converts JDBC calls into the vendor-specified equivalent client API for the specific data system, such as DB2. In order to set up the type 2 driver for DB2, the client libraries must be present at each Java runtime location where the JDBC code is executed.
- The type 3 driver, alternatively described as the open Net Protocol driver, eliminates the type 2 driver's dependence on the native client API. Using type 3 JDBC, a Java client communicates with a network based application server to translate the type 3 protocol requests into database commands.
- The type 4 driver is a purely Java coded socket communications protocol. It comprises a Java layer at the client and at the database server, where the request is translated into database commands.

The type 4 driver has the advantage of requiring no additional database vendor specific client code to be installed at the invoking Java client location. It uses TCP/IP to communicate between client and server.

For use by Java applications with DB2, the IBM DB2 Driver for JDBC and SQLJ implements type 4 behavior. To configure, an applicator repository of Java classes (db2cc.jar plus associated licence) needs to be included in the classpath of each Java client.

In practice, Java client programs, which use the type 4 JDBC driver, need to sit in systems that have access to the following contents of a DB2 9 installation:

`/opt/ibm/db2/V9.1/java`

and

`/opt/ibm/db2/V9.1/lib64`

Providing that a Java client can reference these directory contents, then this Java client may successfully execute JDBC prepared statements and also SQL by using the Type 4 JDBC classes. These directory contents are part of the default installation of any DB2 9 client on the same machine as the application's Java runtime.

The IBM DB2 Driver for JDBC and SQLJ complies with the JDBC 3.0 specification. Additionally it has a number of performance enhancements. On Linux systems, the enhancements are:

- Support to the XML column type, as introduced in the DB2 9
- SSL communication to secure DB2 database servers
- Heterogeneous connection pooling and connection reuse

Connection pooling is the framework for caching SQL connections in JDBC and SQLJ. These are implemented as calls to java.sql.Connection. To minimize the load of making these calls, JDBC 3.0 provides for a factory to make pooled connections to be reused by programs running in participating middleware. WebSphere Application Server is an example of a product that provides a container to implement the pooling.

Connection objects can share a common connection pool, irrespective of differing properties, as supported by the IBM DB2 Driver for JDBC and SQLJ introduced with DB2 Version 9.1. For additional information, please see Java Development with DB2 9 at:

http://www-306.ibm.com/software/data/db2/java/

11.5.2 JDBC driver configuration

A JDBC program begins by invoking the DriverManager class. To accomplish this with the 64-bit Java 5.0, the environment, as defined at the time that the Java runtime was launched, must contain two files in the Java classpath. These are at:

```
/opt/ibm.db2/V9.1/java/db2jcc.jar
/opt/ibm/db2/V9.1/java/db2jcc_license_cu.jar
```

Regarding the database server, we need to tell the middleware (such as an application server) where to find the native shared libraries that implement the type 4 protocol translation. These are at:

```
/opt/ibm/db2/V9.1/lib64
```

In the situation of a stand-alone Java JDBC application, these binaries will need to be in the LD_LIBRARY_PATH of any server hosted JVM. Alternatively they must be specified directly to the participating middleware, such as WebSphere Application Server.

The Java class that implements the type 4 pooling behavior is:

```
com.ibm.db2.jcc.DB2ConnectionPoolDataSource
```

Where there is participating middleware, there is no need for a DB2 client installation at the Java client that invokes the JDBC access to the database. Because the driver is a type 4 driver, it uses TCP/IP to communicate between the Java client and the database server.

11.6 Performance indicators and sizing options

Enhancing the performance of an application requires an understanding of the components, APIs, and runtime service. In this chapter, we have taken the example of Java and reviewed the available options for:

- The JDBC driver
- The JDBC access API
- The 64-bit instance

For tuning in these specific areas, the developer is encouraged to look at the current publications dealing with Java 5 and with DB2 9 performance.

Using the System p virtualization environment, we may create LPARs running Linux ready to be scaled up or down to meet the envisaged requirements of the database instance.

This consideration needs to be fed back to the planning stage of DB2 configuration. However, if the LPAR is found to be underperforming (that is, perpetually stressed) or overperforming (that is, never stressed), then there is the capability to resize the LPAR accordingly.

We may practically divide the requirements of a DB2 instance running in a Linux LPAR between:

- Read-mainly stress

  Concurrent read access creates high demand and may lead to a potential bottleneck if the instance is poorly resourced.

- Write-mainly stress

  Concurrent or sequential database insertions and updates may overwhelm the instance and create delays if the instance is poorly resourced.

Both scenarios offer scope for tuning and workloads are often divided between those issuing complex queries and those that are high volume high concurrent demand. However, a planning decision that needs to be right initially is to understand the processor allocation to LPARs sufficient to build the database.

To discover the trends in the ability of programs to stress DB2 instances, we investigated the performance of a DB2 instance driven from a JDBC program to build a database of just greater than 1.2 million records. In the particular scenario, the database is always built from scratch.

The table space organization for the example database is given in Example 11-4. It shows an example database with a simple division into four separate table spaces,

Example 11-4 Table space organization for a database created with Data and Logs on RAID arrays

```
[db2inst1@rhel4u4-720-lp1 ~]$ db2 list table spaces show detail

          Table spaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = Database managed space
 Contents                             = All permanent data. Regular
tablespace.
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 16384
 Useable pages                        = 16380
 Used pages                           = 9836
 Free pages                           = 6544
 High water mark (pages)              = 9836
 Page size (bytes)                    = 4096
 Extent size (pages)                  = 4
 Prefetch size (pages)                = 4
 Number of containers                 = 1

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x0000
   Detailed explanation:
     Normal
```

```
 Total pages                             = 1
 Useable pages                           = 1
 Used pages                              = 1
 Free pages                              = Not applicable
 High water mark (pages)                 = Not applicable
 Page size (bytes)                       = 4096
 Extent size (pages)                     = 32
 Prefetch size (pages)                   = 32
 Number of containers                    = 1

 Tablespace ID                           = 2
 Name                                    = USERSPACE1
 Type                                    = Database managed space
 Contents                                = All permanent data. Large
tablespace.
 State                                   = 0x0000
   Detailed explanation:
     Normal
 Total pages                             = 49152
 Useable pages                           = 49120
 Used pages                              = 45760
 Free pages                              = 3360
 High water mark (pages)                 = 45760
 Page size (bytes)                       = 4096
 Extent size (pages)                     = 32
 Prefetch size (pages)                   = 32
 Number of containers                    = 1

 Tablespace ID                           = 3
 Name                                    = SYSTOOLSPACE
 Type                                    = Database managed space
 Contents                                = All permanent data. Large table
space.
 State                                   = 0x0000
   Detailed explanation:
     Normal
 Total pages                             = 8192
 Useable pages                           = 8188
 Used pages                              = 148
 Free pages                              = 8040
 High water mark (pages)                 = 148
 Page size (bytes)                       = 4096
 Extent size (pages)                     = 4
 Prefetch size (pages)                   = 4
 Number of containers                    = 1
```

```
Tablespace ID                        = 4
Name                                 = SYSTOOLSTMPSPACE
Type                                 = System managed space
Contents                             = User Temporary data
State                                = 0x0000
  Detailed explanation:
    Normal
Total pages                          = 1
Useable pages                        = 1
Used pages                           = 1
Free pages                           = Not applicable
High water mark (pages)              = Not applicable
Page size (bytes)                    = 4096
Extent size (pages)                  = 4
Prefetch size (pages)                = 4
Number of containers                 = 1
```

Let us now consider performance outcomes when we use a database configured as shown in the example running against a Java JDBC application designed to simulate stock quote information being input into the database.

11.7 Database guidance for performance on LPARs running Linux

An example database that is performing well at an entry level might be configured along the following lines.

We create at least two RAID 5 volumes striped at 16 KB and present them as separate file systems, /Data (or /Data1, /Data2, and so in, if there are many) and /Logs.

The database may be created as part of a single complete CREATE DATABASE statement or as a short incomplete, initial statement on which the table definitions will be added later. Our initial statements are:

```
db2inst1 > db2 CREATE DATABASE TRADE6 ON /Data
db2inst1 > db2 update db cfg for TRADE6 using NEWLOGPATH /Logs
```

The database schema may be expressed separately as a DDL file for which the CREATE TABLE statement is analyzed carefully into the required number of table spaces and to achieve the correct initial use of SMS or DMS or automatic

storage mapping. Once the DDL (here Trade6.ddl) exists, it may be run so as to create the necessary table spaces for data and indexes:

```
db2inst1 > db2 -tvf Trade6.ddl
```

We have assumed that the table spaces are all at the 4 KB page size.

In relation to a given unpartitioned (or single-partitioned) database configuration, we consider how the settings are affected by:

- The LPAR settings available
- The processors supplied to the Linux system image in each LPAR
- The memory supplied with each LPAR
- The I/O scheme for reading and writing
- The file system (/Data) to hold data

11.7.1 The database configuration

The settings in Example 11-5 need to be checked in the database environment. They are exemplary for Linux on System p running DB2 9 databases.

Example 11-5 Database configuration

```
db2inst1> db2set -all
[i] DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
[i] DB2LINUXAIO=YES
[i] DB2COMM=TCPIP
[i] DB2_PARALLEL_IO=*[
[g] DB2SYSTEM=linux1-db-lp1
[g] DB2INSTDEF=db2inst1
[g] DB2ADMINSERVER=dasusr1
```

The settings

```
DB2_PARALLEL_IO=*
```

and

```
DB2LINUXAIO=YES
```

are not the default settings after installation. These must be made manually by the database administrator upon creation of the database instances (here, db2inst1) for the database(s). In the case of Linux systems using external storage, including high performance storage, the settings have a beneficial performance impact.

In our scenario, we also introduce one new buffer pool appropriate to our 4 KB page size table spaces and size it according to the physical memory available to our LPAR. For convenience, we can set its size initially to AUTOMATIC:

```
db2inst1 > db2 “create bufferpool BP4K immediate size automatic
pagesize 4K"
```

Next, we associate the buffer pool just created with the USERSPACE1 table space:

```
db2 "alter tablespace userspace1 prefetchsize automatic bufferpool
BP4K"
```

In addition to associating the buffer pool BP4K, we have also altered the regular table space that holds our TRADE6 data; we recommend one further change. Using the following command, the table space now respects the stanza NO FILE SYSTEM CACHING:

```
db2inst1 > db2 "alter tablespace userspace1 no file system caching"
```

We have now completed a number of basic tuning steps. The initial configuration is now ready to test in order to assess the performance variables involved.

11.7.2 The LPARs created

For our measurements we have been looking at two identically configured LPARs running Red Hat Enterprise Linux 4 update 4 Advanced Server on one LPAR, and SuSE Linux Enterprise Server 10 on the other.

Using a VIO server, each LPAR may be configured either to dedicate resources or to share them for efficient utilization. The LPARs each may allocate up to three physical cores and access to a common virtual SCSI adapter to an array of RAID 5 drives along with up to 8 GB of main memory per LPAR.

11.7.3 Number of processors allocated by a database server LPAR

In sizing the LPAR, a practical consideration is how its resources may impact the ability of the TRADE6 database to grow. We shall take the example of when it is being populated with data that emulates the users of a stock quotation facility and the shares they may receive. The program to generate this data is supplied as part of the Trade 6 Performance Testing Suite. We consider its configuration when driving insertions into the TRADE6 database, directly as a Java client process through JDBC.

When we look at how fast the database may get populated, we need to impose a representative end target on this assessment. We have taken, as our example, the speed at which the database may grow from zero records to just greater than 1.2 million records in the entire database.

The advantage of this statistic is that various tests indicate that the resulting time measurement was repeatable to within 3% on each example run.

Figure 11-2 shows the measured speed of insertion of database rows in the example database compared against the number of physical cores that are configured into the LPAR running under SUSE Linux Enterprise Server 10.

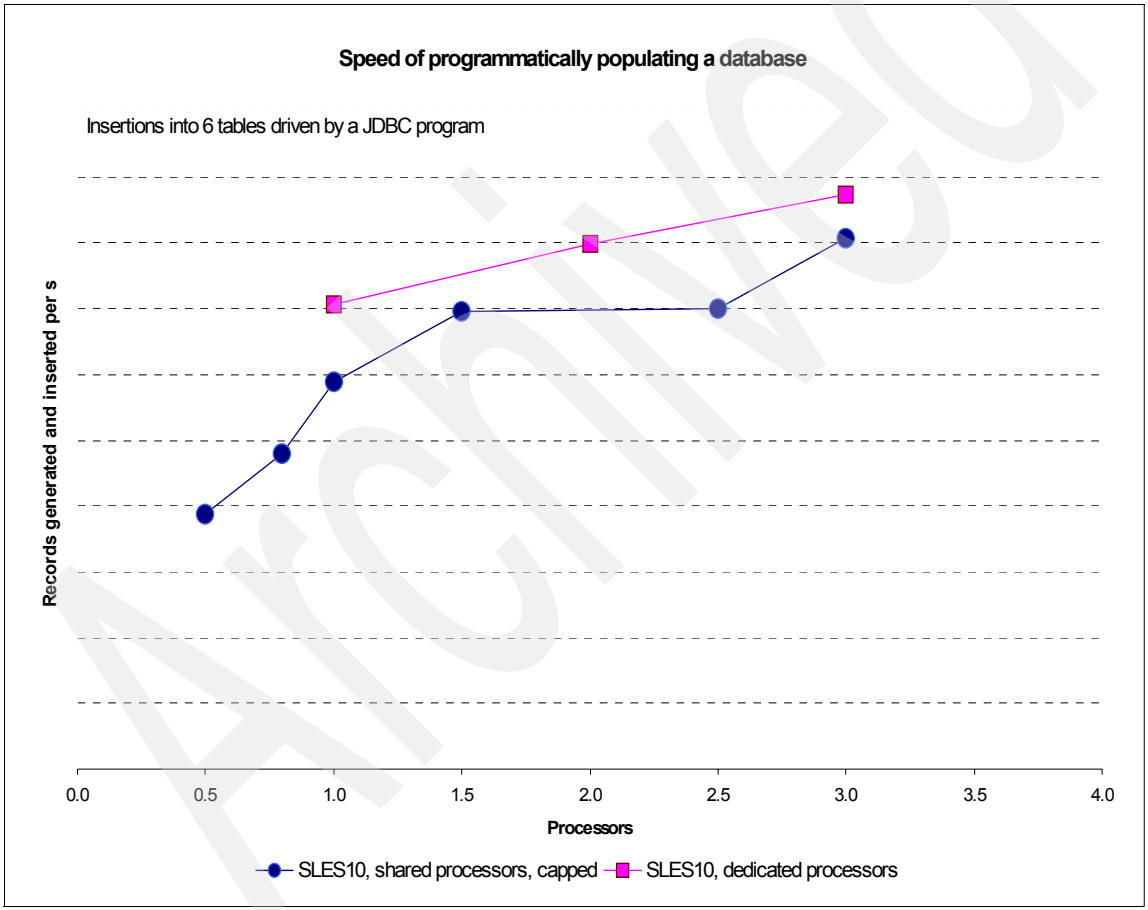


Figure 11-2 Speed of insertion of 1,201,421 rows into database TRADE6 using one SUSE Linux Enterprise Server 10 LPAR with shared versus dedicated virtual processors

The LPAR is devised to contain the minimum number of virtual processors possible for the physical cores allocated. This means that the number of virtual processors provided in each measurement is as selected in Table 11-2. This is in agreement with the suggestions we made in 3.3.3, “Micro-partitions” on page 56.

Table 11-2 Virtual processors configured as a function of fractional number of physical cores chosen

Proportion of physical cores allocated	Virtual processors presented by LPAR
0.10 to 1.00	One virtual processors
1.01 to 2.00	Two virtual processors
2.01 to 3.00	Three virtual processors
3.01 to 3.20 (3.20 is maximum available on the server under test)	Four virtual processors

Figure 11-3 shows the same comparison using Red Hat Enterprise Linux 4 update 4.

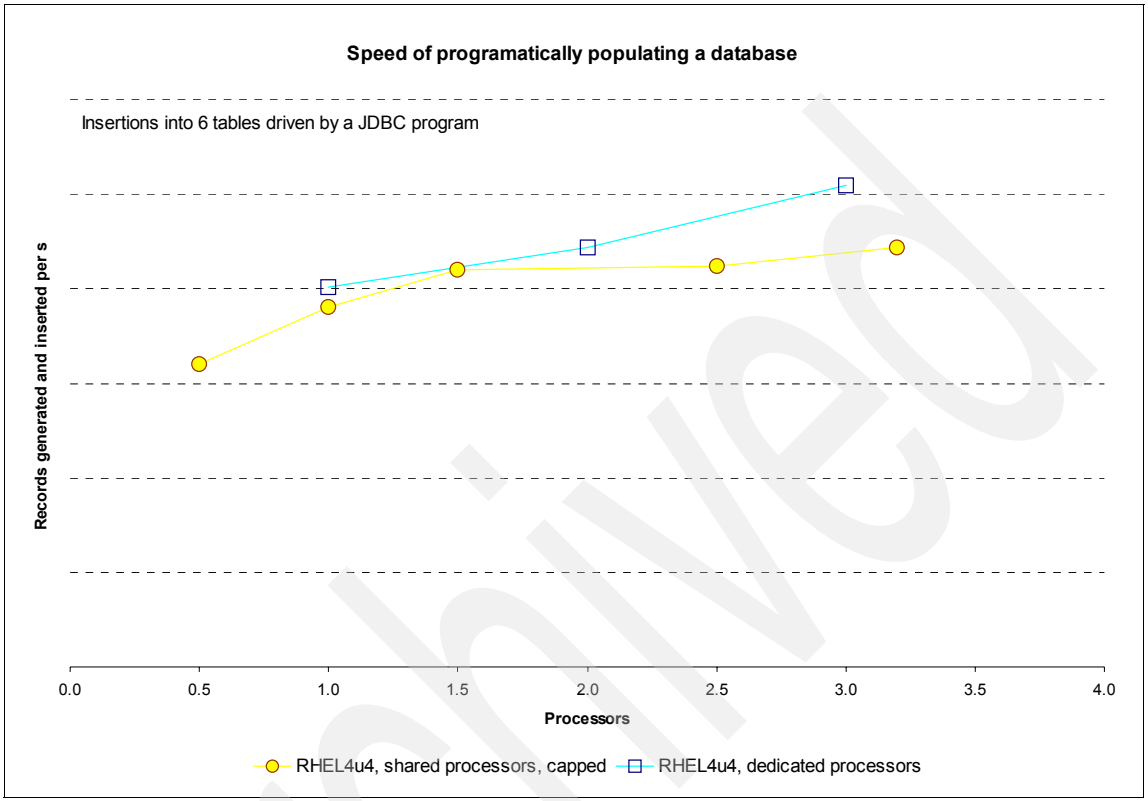


Figure 11-3 Speed of insertion of 1,201,421 rows into database TRADE6 using one Red Hat Enterprise Linux 4 update 4 LPAR with shared versus dedicated virtual cores

If the number of virtual processors was increased above the values shown in Table 11-2 on page 378, then performance degradation is observable in our example scenario. In the worst case of allocating an extra virtual processor to a single physical core LPAR, we observed greater than 10% worsening of the database insertion speed.

By inspection of the measurements as charted, there are two trends:

- Dedicated processors

  See the square shaped (□) points plotted on Figure 11-2 and Figure 11-3. The curve of dedicated processors against the rate of insertion of records into the growing database is close to linear. Within the range considered, each extra core adds about 11% to 12% speed of insertion increase. This is shown by looking either at the trend on the Red Hat Enterprise Linux LPAR or at the SUSE Linux Enterprise Server LPAR.

- Shared processors

  See the circular shaped (○) points plotted on Figure 11-2 and Figure 11-3.

  The curve where the processors are shared is not linear.

  There is some steady growth in the ability to insert records at greater than 1.5 cores, although the curve does not climb noticeably until greater than two virtual processors. Conversely, the rate of population of the database falls away below 1.5 cores and becomes CPU starved at around 0.8 cores.

11.7.4 Memory available to each LPAR

Decreasing the main memory allocated to each LPAR made no significant impact on the database population speed example. The crucial factor was that sufficient memory could be set aside for buffer pools. Our estimate of memory needed was 30%, although much higher percentages are obtainable.

The crucial decision point is at what level we decide that the LPAR is purpose specific to database work and can only be used as a database server. If this is a safe assumption, then the amount of memory set aside should be a safe proportion of instance memory. An amount such as 70% of available memory for buffer pools could be chosen.

11.7.5 I/O subsystem settings

For database systems that are doing their own file systems allocation, as in the case of DMS and automated storage designs of table space, then the Linux file system is liable to do unnecessary work. There are two factors that require compensation:

- Unnecessary caching of files in the file system's page buffer cache
- Unnecessary journaling by the file system

The extent of potential for compensation depends on the file systems used.

Asynchronous I/O

In Linux, asychronous I/O (AIO) allows application threads to continue useful processing without waiting for their I/O requests to complete. AIO provides one interface through which the application may submit its I/O calls and a separate interface for it to obtain the results from its completion group. The advantage to DB2 databases is that we can speed up prefetching of pages.

AIO is a standard package distributed with the 2.6 Linux kernel. Providing the Linux system shows libaio-0.3.96 or later in its list of RPMs, which means it is running a kernel that supports AIO.

In the V9.1 release of DB2, we found that setting the profile registry variable DB2LINUXAIO to YES resulted in some performance gain. We used SystemTap to verify that AIO reads and writes were occurring only, providing that the DB2LINAIO flag was set to YES. See 5.1, “Universal system monitoring tool: SystemTap” on page 88.

Direct I/O

The default behavior of Linux file or block device systems is buffered I/O. By introducing a buffer, the file system (or, analogously, a block device) caches data that is read from or written onto disk. A typical read operation involves physical disk access to read the data from disk into the file system cache, and then to copy the data from the cache to the application buffer. Similarly, a write operation involves physical disk access to copy the data from the application buffer into the file system cache, and then to copy it from the cache to the physical disk.

It is advantageous for a DB2 database to switch off the buffered I/O behavior of Linux, so that it can bypass any layer and access disk directly without any other form of cache. This presumes that a sufficient buffer pool has been set up for its own in-memory caching to occur in place of that of the operating system.

Direct I/O in DB2 9 can be set up for two storage management options:

- Using DMS whose containers are raw block devices, such as contiguous storage on a dedicated device (such as /dev/sdd, if this fourth disk device were an external disk system)
- Using DMS whose containers are mounted files (such as /Data in our example).

These options were reviewed in our earlier discussion, “Table spaces” on page 353.

Direct I/O is exploited by DB2 using the NO FILESYSTEM CACHING clause in the definition of a table space.

The combined effect of these settings was an uplift of just greater than 12% in the best case using our example database to accept one million inserts and updates.

11.7.6 The file system

Considering our simple JDBC program running concurrently to insert one million records, the result of the AIO and Direct I/O optimizations were dependent on the file system. For our particular test case, the changes are summarized in Figure 11-4.

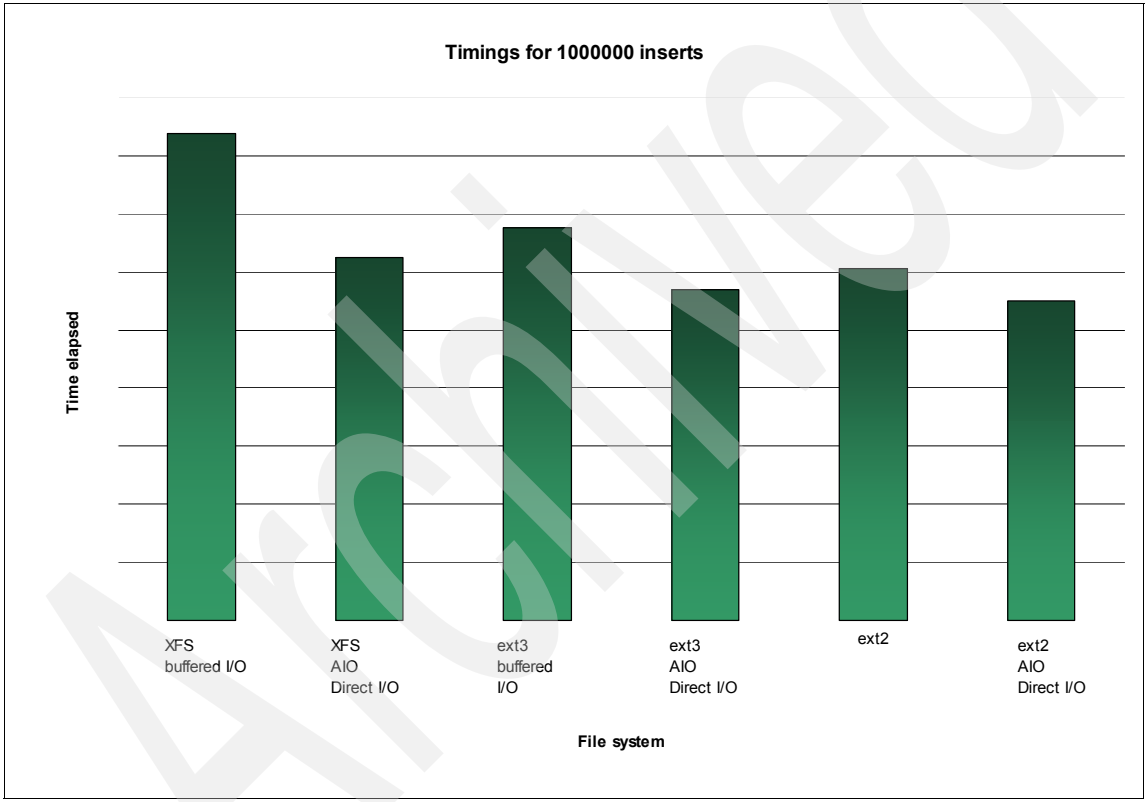


Figure 11-4 Impact of AIO and Direct I/O settings variation with file system

The dominance of ext2 is largely explicable by the fact the I/O buffering is a significant load in a file system using DMS or automatic storage. Good results were also obtained from the other file systems, provided that DB2LINUXAIO was set to YES and the FILE SYSTEM CACHING OFF clause was issued on the table space used by the regular data.

11.8 Wizard guided tuning

We have looked at initial set up conditions and some limited tuning steps that have demonstrated performance benefits and trends. For the administrator wishing to tune a database comprehensively, they have the option of using the visual user interface facilities provided on many platforms to tune DB2. These are often found to produce useful guidance.

For our example, we are running the DB2 9 Control Center on Linux x86. The code for launching the Control Center may be found at:

/opt/ibm/db2/V9.1/bin/db2cc

Any instance user account[2] can launch the tool by invoking db2cc. The GUI, shown in Figure 11-5, allows a selection of tools and wizards useful for the entire database life cycle. For performance tuning, our focus is on the Configuration Advisor.

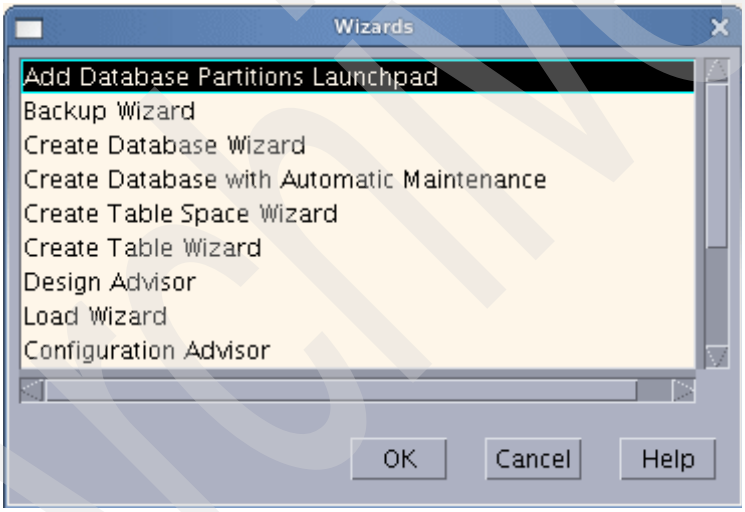


Figure 11-5 The DB2 9 Control Center user interface showing available wizards

The Configuration Advisor is a straightforward tool for making tuning decisions. It has seven steps:

1. Locate the existing database to be tuned.

[2] On supported platforms. These include Linux x86 and AIX 5L V5.3.

2. Decide upon the proportion of RAM that goes to the Database Manager Global Memory, as in Figure 11-6.

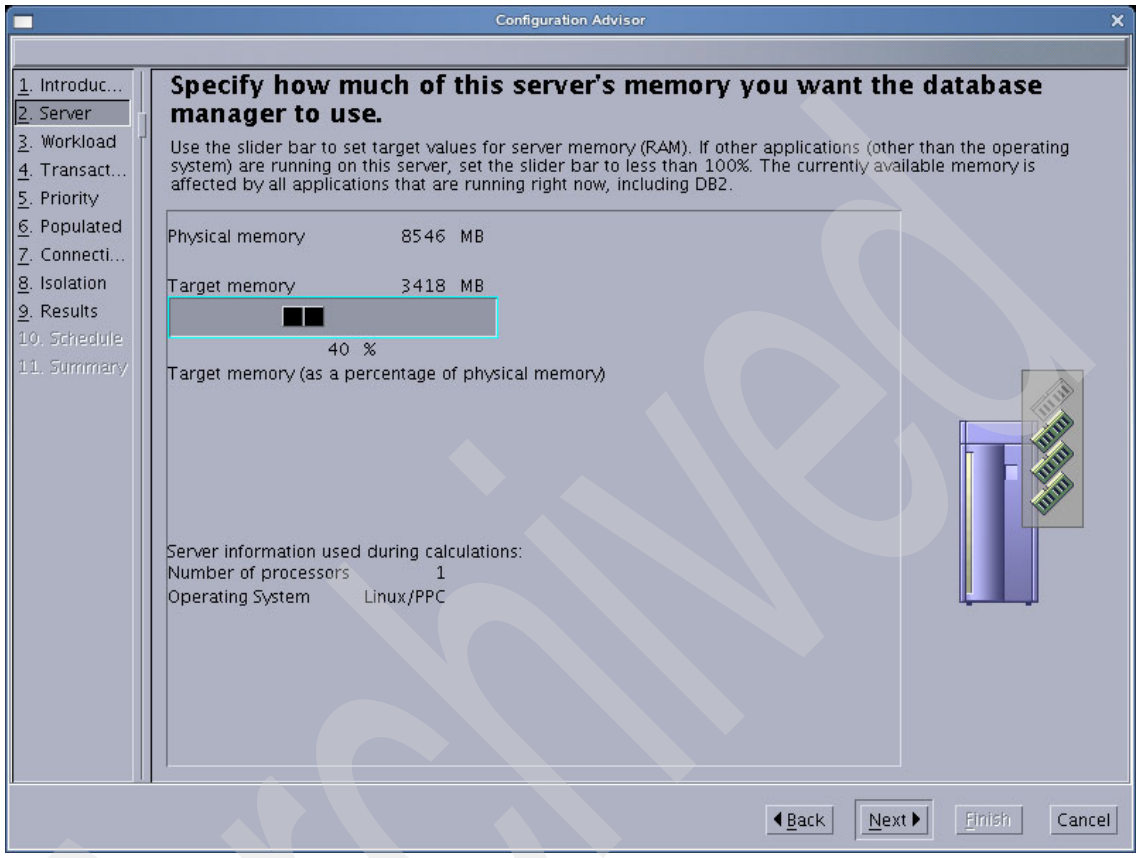


Figure 11-6 The DB2 9 Configuration Assistant to determine the proportion of memory available for the Database Manager Global Memory

3. Characterize the workload as Online Transaction Processing, as Data Warehousing, or a mixture.
4. Characterize a unit of work in terms of how many statements on average before a commit.
5. Optimize the wizard for either fast recovery or fast performance.
6. Input the expected number of local network and remote online connections; it may be useful to consider the peak load.
7. Select the isolation level for executing statements to determine the likelihood of dirty or phantom reads interfering with the correct running of the application.

Figure 11-7 a review of the wizard's recommendations.

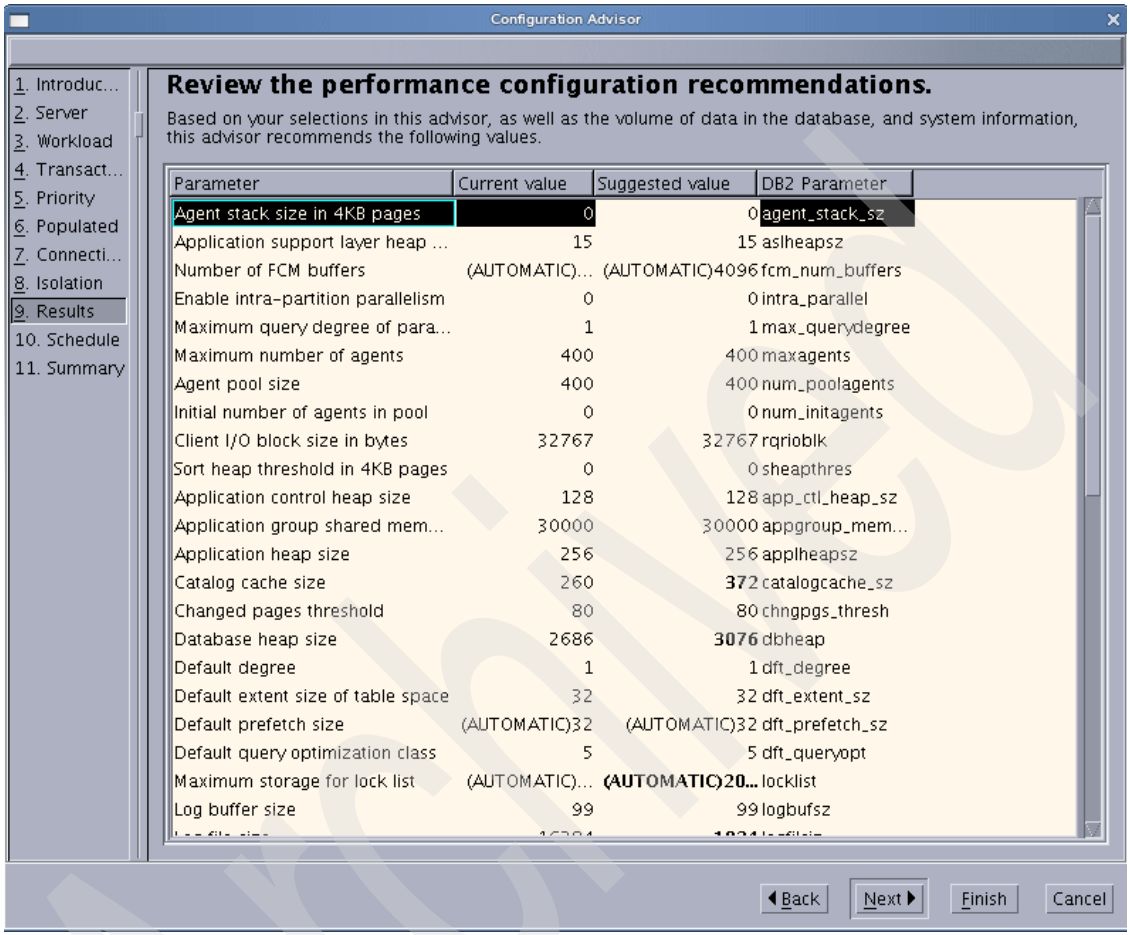


Figure 11-7 The DB2 9 Configuration Assistant review of performance recommendations for tuning the database

This is followed by a detailed breakdown of all the actions to be automated by the wizard, either immediately or at a scheduled time, as shown in Figure 11-8.

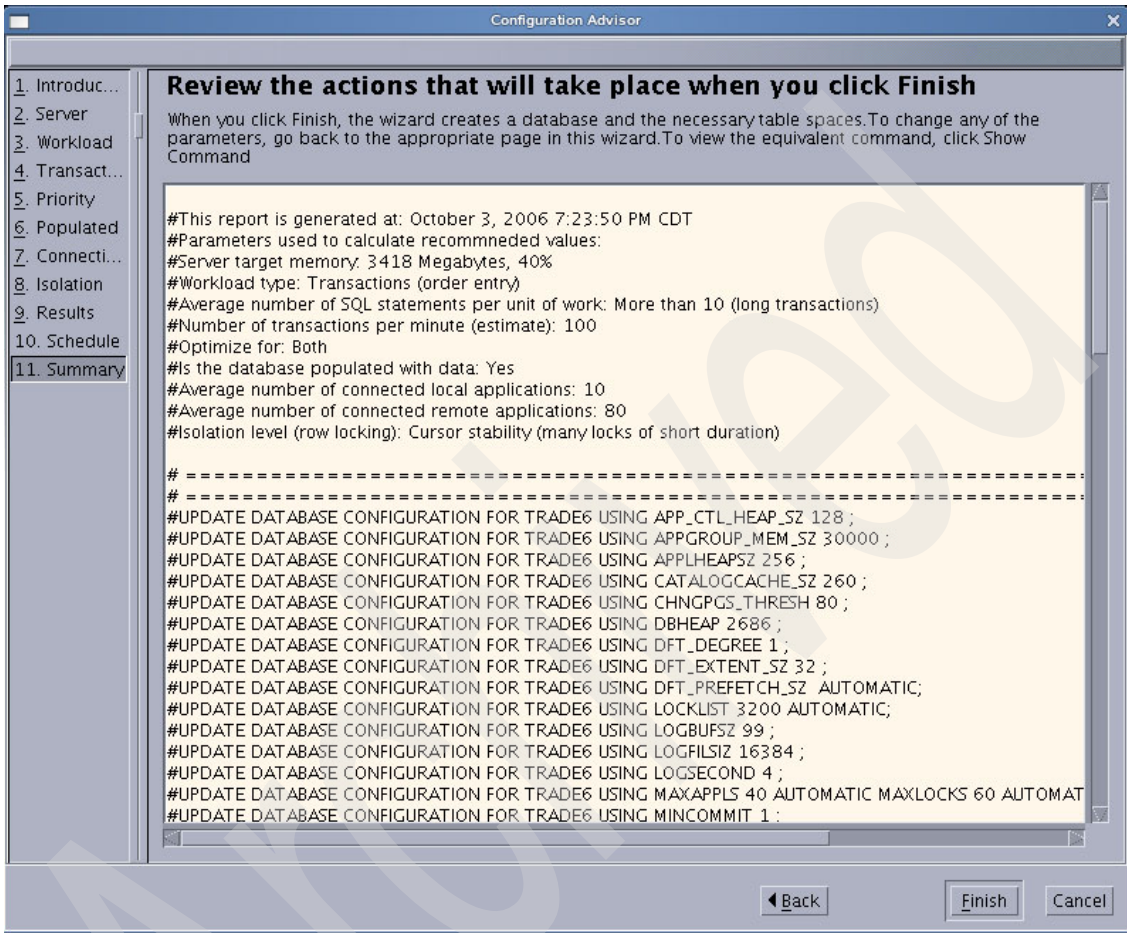


Figure 11-8 The DB2 9 Configuration Assistant shows the adjustments to be made to the database configuration.

11.9 Database guidance for DPF

DB2 Enterprise 9 data server permits parallel execution both within a single partition, by setting up intra-partition parallelism, and between partitons. To achieve the multi-partition set up, we need DPF.

A partitioned database environment is a database that is made up of two or more database partitions. In this type of database, data is hashed for storage. A database partition consists of its own data, indexes, configuration files, and transaction logs.

Tables can be located in one or more database partitions. When a table's data is distributed across multiple partitions, some of its rows are stored in one partition, and other rows are stored in other partitions. Data retrieval and update requests are decomposed automatically into sub-requests, and executed in parallel among the applicable database partitions. The fact that databases are split across database partitions is transparent to users.

Typically, a single database partition exists on each physical component that makes up the computer. The processors on each system are used by the database manager at each database partition to manage its part of the total data in the database. Because data is divided across database partitions, we can use the power of multiple processors on multiple computers to satisfy requests for information.

Data retrieval and update requests are decomposed automatically into subrequests and are executed in parallel among the applicable database partitions.

Using DPF, an updatable partitioning map is used with a hashing algorithm to specify the mapping of partitioning key to database partitions, which is used to determine the placement and retrieval of each row of data.

A workload can then be spread across multiple partitions for large tables, while allowing smaller tables to be stored on one or more database partitions. Since each database partition has local indexes on its data, there is increased performance for local data access.

DB2 also supports partial de-clustering, where tables and table spaces can be spread across a subset of the available partitions. Depending on the number of database partitions, we may have one or more single-partition database partition groups, and one or more multi-partition database partition groups. Each partition must use a unique partition number and the same database partition may be found in one or more database partition groups.

To ensure fast recovery of the partition containing system catalog tables, avoid placing user tables on the same database partition. This is accomplished by placing user tables in database partition groups that do not include the database partition in the IBMCATGROUP database partition group.

The partition number is automatically appended to the path. This is done to maintain the uniqueness of the path in multiple logical partition configurations.

The Design Advisor is a convenient visual tool for designing partitioned databases. It can be launched by running the **db2advis** command.

We recommend the following items:

- Small tables should be placed in single-partition database partition groups, except when the application can take advantage of collocation with a larger table. Avoid extending medium-sized tables across too many database partitions. For example, a 100 MB table may perform better on a 16-partition database partition group than on a 32-partition database partition group.
- Database partition groups can be used to separate online transaction processing (OLTP) tables from decision support (DSS) tables, to ensure that the performance of OLTP transactions are not adversely affected.
- In a multi-partition database partition group, we can only create a unique index if it is a super-set of the partitioning key.
- When creating a database, make sure to specify a local (not shared) directory in the “ON” clause for the database location. For example, CREATE DATABASE SAMPLE ON /Data, where /Data is a pre-existing local directory.
- The default database path (DBTDBPATH) parameter in the database manager configuration defaults to the location of the home directory of the instance owner (which is shared by NFS). When creating a database without specifying the database location, it will create a database using DBTDBPATH, which points to the shared instance-owner directory. This will degrade performance.
- Once the database is created, we should ensure that each data partition should also have its own log local directory. This is achieved by the following command:

  ```
  db2 update db cfg for SAMPLE using NEWLOGPATH /Logs
  ```

11.10 Summary

We have looked at the performance of DB2 9 databases running under Linux on System p hardware. In a virtualized environment, the performance factors that have been critical are virtual processor allocation, the speed of the hard disk, and the I/O subsystem settings.

Both memory and the hard disk can be tuned at the database planning stage and settings for DB2_PARALLEL_IO for table spaces and adding buffer pools as memory permits can be done.

The possibility of running one or more databases under dynamic LPAR changes was also investigated, along with micropartitioning. The finding was that speed of execution is seriously impaired when less than a single core was allocated as a single virtual processor. However, above this limit, there are possible gains of adding a fraction of a core, provided that there are a minimum number of virtual processors for the cores allocated.

Tuning the I/O systems on Linux carries significant benefits. In particular, the setting of DB2LINAIO to YES needs to be a continued practice (originally introduced in the V8.2 release). The clause FILE SYSTEM CACHING OFF applied to table spaces of the database's regular data is also significant. Benefits were reaped from these settings on all file systems.

The DB2 9 data system is a 64-bit architecture, which we have tested under 64-bit configuration on the Red Hat Enterprise Linux 4 update 4 and SUSE Linux Enterprise Server 10 releases of Linux for System p. The properties of the 64-bit Java environment to access DB2 9 databases and drive a load to it have also been reviewed.

To maintain detailed tuning steps, the Control Center's Configuration Advisor wizard has been reviewed and steps to tune an unpartitioned database or to scale up to a DPF database environment have been examined.

12

WebSphere Application Server V6.1

This chapter examines performance factors that impact Java enterprise applications running within IBM WebSphere Application Server V6.1 under Linux on System p. We shall be looking at steps that we may take to maximize the ability of Application Server instances to satisfy demand from Web-based requests submitted near simultaneously across the internet or across an intranet. Additionally, we shall look at the possibilities of achieving better balance of workloads by changing the number of logical partitions (LPARs) that host multiple Application Server instances.

To understand the enterprise application workloads that we may wish to achieve, first we review:

- The programming model for Java applications supported by WebSphere Application Server V6.1.
- The role of dynamic content in Java Web applications.
- The extended programming model for Service Oriented Architecture (SOA) applications supported by WebSphere Application Server V6.1.

WebSphere Application Server is optimized for multithreaded applications developed according to the Java 2 Enterprise Edition specification. Additionally, we can sometimes get significant performance improvements through scaling up the number of places where the Java application is running.

The ability to scale one or many applications, across multiple physical or logical servers is an objective satisfied by the IBM WebSphere Application Server Network Deployment (ND) product. We shall review how best to set up the Linux system to support either single server or distributed Java applications in System p environments by:

- Linux system tuning
- Application environment tuning

To see how a Java application running inside the WebSphere application server runtime can be assessed, we shall run an example, Trade application, and subject it to a load. The load will simulate near simultaneous Web requests originating from multiple users.

12.1 Java 2 Enterprise Edition

The programming model for applications to be deployed on WebSphere Application Server V6.1 is based on Java and open source specifications. A significant specification is the Java 2 Enterprise Edition. This was originally formulated to provide a standard for creating robust, distributed applications that can be made resilient against localized failure in the system or in the network.

The Java 2 Enterprise Edition (J2EE) standard sets out the characteristics of an environment for executing application-level components. In the J2EE 1.4 release of this specification, application-level components include Web components and Enterprise Java Bean (EJB™) components. Web components, such as Java Servlets and Java Server Pages, provide dynamic responses to requests from a Web page. EJB components contain server-side business logic for enterprise applications.

See Figure 12-1 for a list of the industry-standard APIs supported.

```
Java Servlet Specification 2.4
Java Server Pages Specification 2.0
Enterprise Javanese Specification 2.1
Enterprise Javanese to CORBA Mapping 1.1
RMI over IIOP
Java IDL API
Web Services for J2EE, Version 1.1
SOAP with Attachments API for Java Specification 1.2
Java API for XML Processing Specification 1.2
Java API for XML Registries Specification 1.0
Java API for XML-based RPC Specification 1.1
JDBC Specifications, 3.0, 2.1, and Optional Package API (2.0)
Java Connector Architecture (JCA) 1.5
Java Message Service Specification 1.1
JavaMail API Specification 1.3
Java Authorization Contract for Containers 1.0
Java Naming and Directory Interface Specification 1.2.1
Java Transaction API Specification 1.0.1B
Java Transaction Service Specification 1.0
Java Beans Activation Framework Specification 1.0.2
```

Figure 12-1 The Java 2 Enterprise Edition 1.4 Specification at a glance, implemented in the WebSphere Application Server V6.1 product set

12.1.1 The structure of J2EE applications

When Web components and EJB components are assembled to be ready for execution, they are placed in Web modules and EJB modules respectively.

WebSphere Application Server V6.1 is a J2EE 1.4 application server product. This implies that it provides runtimes for executing J2EE 1.4 applications.

J2EE applications consist of components, containers, and services. Containers represent the runtime environment of each application server and provides it with the ability to be configured. Web and EJB component containers host services that support Web and EJB modules.

A J2EE application architecture supports component-based development of multi-tier applications. These tiers are typically:

- Presentation front-end tier

  The presentation front-end tier controls the client interactions with the server. This tier is where the Web components, such as Servlets and JSPs, are hosted as part of the J2EE Web container. Alternatively, this tier may consist of stand-alone Java applications that provide a dynamic interface to the middle tier.

- Application logic tier

  In the middle tier or application logic tier, enterprise beans and Web Services encapsulate reusable, distributable business logic for the application. These middle-tier components are contained on a J2EE Application Server, which provides the execution environment for these components to perform actions and hold processing data.

- Enterprise data tier

  In the data tier, the enterprises' data is stored and held persistently, typically in a relational database or relational databases.

  The database itself needs to be tuned in relation to the intended workload supplied by the Java application. Our earlier reviews of the Oracle 10g database, in Chapter 10, "Oracle Database" on page 299, or DB2 9 data system, in Chapter 11, "DB2 9 data server" on page 347, indicated how this should be pursued.

  These relational database management products are candidates for holding the enterprise data. Various relational databases may be used with IBM WebSphere Application Server to implement the data tier of J2EE applications.

For more information about J2EE architecture and the application structure it defines, the reader is encouraged to download and read the J2EE 1.4 Specification. It is available for download at:

http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf

For the readers who need to pursue the development of J2EE programs and the practical steps to deploy the technology via WebSphere, the reader is recommended to refer to *Experience J2EE! Using WebSphere Application Server V6.1*, SG24-7297.

12.1.2 WebSphere Application Server in relation to J2EE

The WebSphere Application Server V6.1 product is an implementation of the J2EE 1.4 specification, which allows conforming business logic programs to run in a cross-platform portable, scalable setting.

This product allows one or more Application Server instances[1] to be launched for the purpose of running business logic programs. Each Application Server instance is implemented as a single Java virtual machine (JVM).

Any JVM, including a JVM that conveys an application server, runs inside the Linux system image as a process.

Application servers use Java technology to augment HTTP Server capabilities to handle Web application requests. An application server makes it possible for a server to generate a dynamic, customized response to a client request.

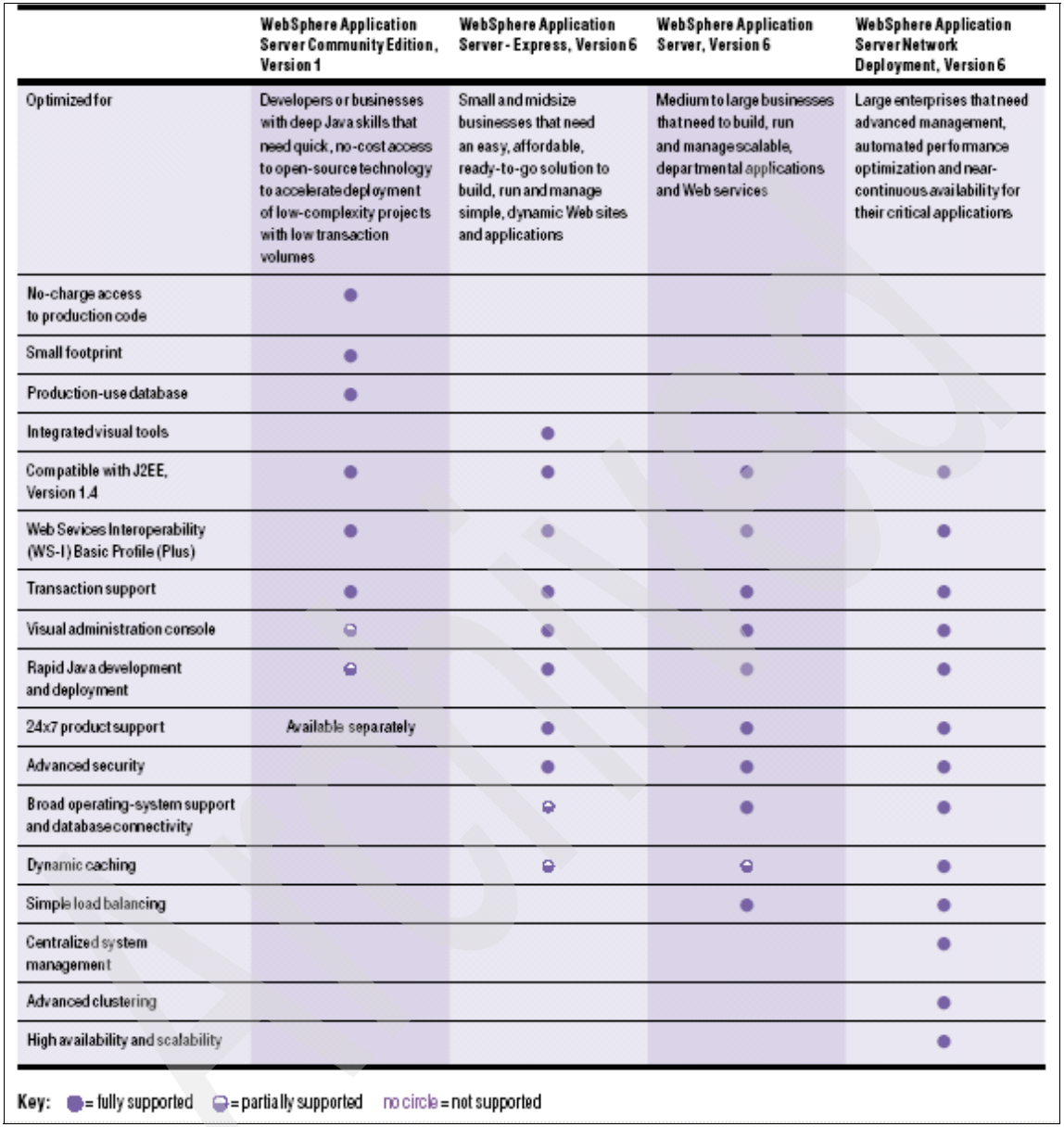
The WebSphere Application Server product provides application servers available to run on each location where it is installed. A location may be a physically separate machine or a logically separate partition.

Where a single enterprise solution has been deployed across many application servers in different locations, we say that the application has been installed on many nodes. A node means the physical machine or logical partition where a group of (one or more) application servers is able to run.

Our focus in this chapter is the WebSphere Application Server V6.1 Network Deployment (ND) product. This product offering is considered because it

[1] It is usually convenient to shorten the expression Application Server instance. When speaking in the context of a distributed J2EE application architecture, we can speak of each such instance as being an application server
An application server is a single managed Java process that is capable of executing Java enterprise applications. There may be one or more application servers executing the same Java application working on one machine, and there may be one or more executing on many machines. If it is clear from the context that we do not mean an Application Server product, such as the WebSphere Application Server ND offering, then one application server means one single instance.

provides for distribution of appserver nodes across multiple machines or multiple partitions. In a Linux on System p setting, we are particularly interested in the tuning opportunities it presents because of the configurable LPAR options that may have an impact on how to size and scale practical business deployments of Java enterprise solutions. The WebSphere Application Server V6.1 ND product offering is a member of the IBM WebSphere Application Server V6.1 stack of product offerings, as shown in Figure 12-2 on page 397.



	WebSphere Application Server Community Edition, Version 1	WebSphere Application Server - Express, Version 6	WebSphere Application Server, Version 6	WebSphere Application Server Network Deployment, Version 6
Optimized for	Developers or businesses with deep Java skills that need quick, no-cost access to open-source technology to accelerate deployment of low-complexity projects with low transaction volumes	Small and midsize businesses that need an easy, affordable, ready-to-go solution to build, run and manage simple, dynamic Web sites and applications	Medium to large businesses that need to build, run and manage scalable, departmental applications and Web services	Large enterprises that need advanced management, automated performance optimization and near-continuous availability for their critical applications
No-charge access to production code	●			
Small footprint	●			
Production-use database	●			
Integrated visual tools		●		
Compatible with J2EE, Version 1.4	●	●	●	●
Web Sevices Interoperability (WS-I) Basic Profile (Plus)	●	●	●	●
Transaction support	●	●	●	●
Visual administration console	◒	●	●	●
Rapid Java development and deployment	◒	●	●	●
24x7 product support	Available separately	●	●	●
Advanced security		●	●	●
Broad operating-system support and database connectivity		◒	●	●
Dynamic caching		◒	◒	●
Simple load balancing			●	●
Centralized system management				●
Advanced clustering				●
High availability and scalability				●

Key: ● = fully supported ◒ = partially supported no circle = not supported

Figure 12-2 The range and capabilities of available IBM WebSphere Application Server V6.1 products

12.1.3 The Java 5 runtime

The JVM delivered with the WebSphere Application Server V6.1 product set on System p servers is provided by IBM for running on supported Linux distributions on System p servers. The JVM comprises a runtime system that may be launched from the command line by calling the command **java** and, in addition, **javac**, the Java compiler along with other associated Java 2 System Development Kit (SDK) tools.

The version of the Java 2 SDK implemented by the WebSphere Application Server V6.1 product family conforms to the Java 2 Standard Edition 5 (J2SE™ 5) language specification. See Figure 12-3 for the adopted features.

```
JSR 003: The JMX 1.2 specification. Packages: javax.management.*
JSR 013: Additions to java.math for improved arithmetic operations
using BigDecimal
JSR 028: The Java SASL packages: javax.security.sasl
JSR 114: JDBC Rowset implementations that specify rowset more
completely
JSR 133: Java Memory Model and Thread Specification Revision
JSR 160: JMX remote API, V1.0
JSR 163: Java Platform Profiling Architecture, JVMTI (replaces
JVMPI)
JSR 166: Concurrency utilities. Packages: java.util.concurrent.*
JSR 174: Monitoring and Management Specification for the Java
Virtual Machine
JSR 175: A Metadata Facility for the Java Programming Language
JSR 200: Network Transfer Format for Java Archives
JSR 201: Extending the Java Programming Language with Enumerations,
Autoboxing, Enhanced for Loops and Static Import
JSR 206: Java API for XML Processing (JAXP) 1.3
JSR 204: Unicode Supplementary Character Support
```

Figure 12-3 New features adopted through the Java Community Process as Java Service Requests now incorporated into Java 2 Standard Edition 5

The J2SE 5 virtual machine specification adds several features and functions to benefit application developers. These new features include generics, auto-boxing of primitives, annotations (API documentation metadata for code generation), and support for enumerated types coded as enums. This makes development quicker, easier, and less error prone. For example, generics should help eliminate issues with ClassCastExceptions from items like vectors, because containers based on generics will allow compile-time catching of incorrect assignment or casting.

J2SE 5 is upwards binary-compatible with the previous runtime (J2SE 1.4.2), except for the incompatibilities documented by Sun Microsystems at:

http://java.sun.com/j2se/1.5.0/compatibility.html#binary

Most of the incompatibilities refer to compiling classes at a Java 2 SDK 5 level using a target of 1.5. “Almost all existing programs should run on J2SE 5.0 without modification,” to cite the Sun documentation.

The Java 2 Standard Edition 5 (J2SE 5) language specification may be found at:

http://java.sun.com/j2se/1.5.0/docs/api/index.html

There is a choice of 32-bit and 64-bit JVMs supplied with the WebSphere Application Server V6.1. Installing the latest level of the product is supported on the current Red Hat Enterprise Linux Advanced Server 4 and the current Novell SUSE Linux Enterprise Server 10 releases. The appropriate release to work with the current IBM software products running on Linux on System p is the 64-bit package of IBM WebSphere Application Server V6.1.

Earlier releases of Red Hat Enterprise Linux and Novell SUSE Linux Enterprise Server are also possible. The details of the current requisite levels of Java for the supported operating systems running on Linux on System p may be checked at:

http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007679

At the time of writing this chapter, the latest level of IBM WebSphere Application Server is V6.1.0.2. It it recommended to keep it up to date with Fix Packs. The latest update may downloaded from the support Web site:

http://www.ibm.com/software/webservers/appserv/was/support/

12.2 Application components in the WebSphere runtime

The WebSphere Application Server V6.1 programming model encompasses the provisions of the J2EE 1.4 specification along with extensions for programmer productivity and new capabilities. It allows applications to run that are Java coding of enterprise information services and, additionally, which internetwork with other technologies by using Web Services.

Applications may be deployed to the Application Server whenever they conform to the J2EE specification. Characteristically, this means that the application:

- Runs Java Servlets, Java Server Pages, or Enterprise Java Beans, including instances of a session bean type or an entity bean type or a Message Driven Bean.
- May expose its method calls as Web Services.
- May enlist in transactional services and require transaction coordination between different resources, such as data sources and messaging engines.
- Requires the use of system management tooling, such as Web-based user interfaces and scripting languages, to deploy and administer in the larger arena.

In the case of IBM WebSphere Application Server advanced edition and above, the administrative modes for system managament are:

- A Web-based client user interface titled the Integrated Solutions Console allows interactive configuration of applications server containers, resources, and runtime parameters.
- A scripting language called *wsadmin*, which can be invoked by issuing the command **`wsadmin.sh`**.

  This allows configuration of the same characteristics in a command line invoked shell scripting style that may be suitable for bulk tasks or batch processing.

The purpose of the adminstrative modes is to adminster the application components and the resources that they refer to. Resources may be categorized into:

- JDBC providers
- JMS providers
- J2EE Connector Architecture (JCA) enabled enterprise information systems

These are offered as tools to help the person deploying a software solution to obtain sufficient management of resources to achieve the required level of robust performance. We mentioned, in 12.1, “Java 2 Enterprise Edition” on page 393, that a major purpose of J2EE program environments was the ability to configure for resilience against local failure. One strategic aspect of this is transactional support. In J2EE, most aspects of transactional support are transparent to the Java program developer, but the behavior at transaction boundaries are determined by settings of parameters at the time of deployment. In a J2EE environment, the intention is that the application drives these resources in a transactional way, as shown in Figure 12-4.

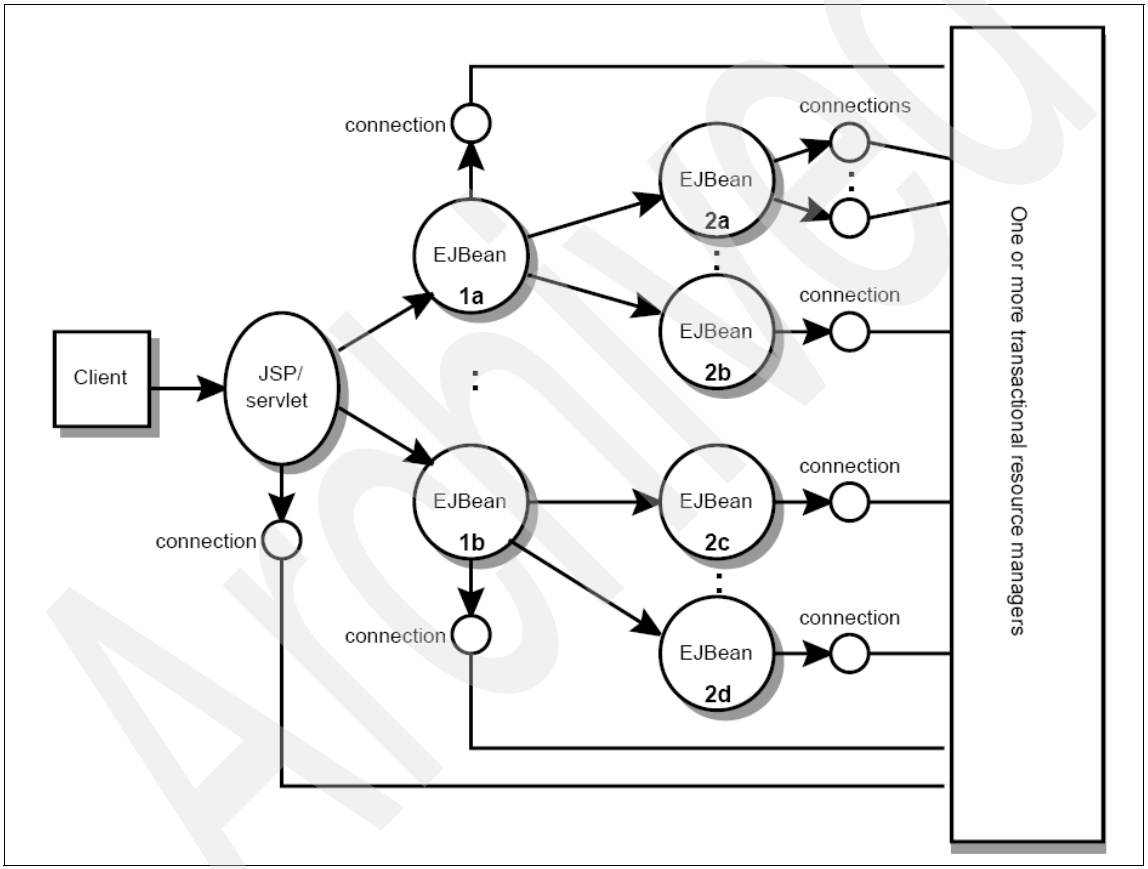


Figure 12-4 Supporting the EJB with transactional resources as illustrated in the J2EE specification

12.2.1 Dynamic content in the WebSphere programming model

Application servers can be used to provide Java Web application capability by using Java technology to augment HTTP Server capabilities to handle Web application requests. An application server makes it possible for a server to generate a dynamic, customized response to a client request, reflecting the business logic characteristics of the application.

Typically, Web requests are directed to one or more HTTP Server instances. An advantage of this division of responsibility in a production setting is that static content may be served by the specialized delivery mechanism of the HTTP Server. Requests for dynamic responses to a user's data entry are directed to the application server's Web container for service.

Note: Static and dynamic Web content

Static content and dynamic content usually needs to be combined together to achieve a single business function.

Static content indicates the unvarying content that can be fetched by simple uniform resource identifier (URI) expressions. Examples of this content are GIF and JPEG image format files. A complete HTML document, complete with all links, down to the HTML end-terminator is also an example of static content. Good performance is usually obtainable by having these fetchable from a filestore, or possibly cached in memory, from a specialized server located as close as possible to the network from which the request for static content is made.

Dynamic content indicates a response that needs to be computed. The response that is returned to the Web client needs to take account of the parameters submitted in the request, the status of the request, such as the server it came from, the informational state found by consulting other tiers, and any programmatically expressible factors that can influence the business decision so as to shape an appropriate response.

In the WebSphere Application Server product, an application server usually works with an external HTTP Server to handle requests for Web applications. The application server and HTTP server communicate using a WebSphere HTTP plug-in. This plug-in is specific to the HTTP Server and platform. For example, there is a plug-in that needs to be selected if the site installation makes use of Apache 2.0 on Linux on System p. See Chapter 9, "Apache Web server" on page 265 for more details on how to configure this HTTP Server for performance. Also, there is a plug-in that needs to be selected if the site installation makes use of IBM HTTP Server V6.1. The available plug-ins for use

in the application server may be selected at WebSphere Application Server product installation time or during administration later if the need arises.

12.2.2 The WebSphere Web container

Provided that the developer has created valid content and deployed the application, then the WebSphere Web container is the place where the programmer's Web module will be executing once the application has started. The developer's Web application needs to be created into a Web module by assembling servlets and JSPs, together with resource references to static content.

The Web module might be a stand-alone application serving a complete business purpose. Alternatively the developer will need to combine it with other modules so as to provide a multi-module J2EE application. In either case, the assembled application will need to be deployed to the application servers.

Note: The structure of Web applications

The two dynamic components defined by J2EE 1.4 for execution in a Web container are Servlets and JSPs.

Servlets can perform such tasks as supporting dynamic Web page content, providing database access, serving multiple clients at one time, and filtering data.

Java Server Page (JSP™) files enable the separation of the HTML code from the business logic in Web pages. IBM extensions to the JSP specification make it easy for HTML authors to add the power of Java technology to Web pages without being experts in Java programming.

At deployment time, there are some key decisions. A key decision that has a significant effect on behavior and performance of the running application is whether to switch on HTTP Session tracking. By default, HTTP Session tracking is switched on in WebSphere Application Server V6.1. Session tracking allows each HTTP response to be associated 1:1 with the HTTP session originating from the unique user at a unique browser who first made the request.

By default, all WebSphere servers have session affinity. This means that the concurrent requests are routed to the application server that serviced the first request. Session affinity provides a significant performance optimization, because it reduces the number of times a server will need to fetch session data, because in the situation where the server continues to have capacity, it will already have stored the user's session data in memory. Therefore, session

affinity relies on keeping session objects in memory local to the application server instance.

Note: Session tracking behavior in a Web container

An HTTP session is a series of requests to a servlet, originating from the same user at the same browser. Managing HTTP sessions allows servlets running in a Web container to keep track of individual users.

Sessions may be tracked automatically on the application server by holding a cookie supplied with the original request. Alternatively, instead of requiring that the client can accept cookies, the application may be designed to use Web address rewriting. With this design, the servlet must be written to include the Web address parameters in the response received by the originating browser.

An additional option in session tracking is to use a Secure Sockets Layer (SSL) encrypted session identifier to relay between the browser and the HTTP Server. This can use either cookies or Web address rewriting. This design is supported for HTTP Servers that include SLL support, including the IBM HTTP Server.

The objective of session tracking is to ensure that the HTTP server properly routes a session back to the same application server for each request. This feature is called session affinity.

If session affinity cannot be relied upon in the application design as the mechanism to provide the recovery of session data, then we need to find other ways of keeping the server nodes in step.

12.2.3 HTTP Session persistence options

We may require that each server in a cluster has the ability to recover session data, such as where session affinity has not been allowed, possibly because session tracking cookies are not allowed, or because we need more robust ability to recover from failure of a single application server. To provide this, we have the possibility of configuring each server node to fetch the most recent session data. The options are:

1. Hold session state in memory, non-persistently, as assumed in the preceding section. This is the default option.

2. Configure the application servers to keep session state written out to a central session database. This needs to be a shared database where there are many application servers providing a common application, as discussed in 12.3, “The building blocks for WebSphere clustering” on page 413.
3. Configure memory-to-memory replication between application servers (in the ND environments discussed in12.3, “The building blocks for WebSphere clustering” on page 413). WebSphere’s internal replication allows sessions to be held in memory with the ability to distribute and recover state without using a database.

12.2.4 The WebSphere EJB container

An EJB module is used to assemble one or more enterprise beans into a single deployable unit. An EJB module is stored in a standard Java archive (JAR) file.

The EJB container provides all of the runtime services needed to deploy and manage enterprise beans. The container runs as a server-side process that handles requests for all types of enterprise beans defined in the application.

Note: File formats used in Java applications

A JAR file (or Java ARchive) is a platform-independent compressed file format used to distribute a set of Java classes. It is used to store compiled Java classes and associated metadata.

Web Application aRchive (WAR) files are also Java archives that store XML files, Java classes, servlets, and JSPs for Web applications.

Servlet Application aRchive (SAR) files store XML, classes, and servlets. They are functionally equivalent to WAR files, if the Web application has no JSPs. The current practice is that they most commonly are used to hold SIP servlets, which are described in 12.2.6, “WebSphere Application Server V6.1 product architecture” on page 410.

An EJB JAR file utilizes the JAR file specification. It contains enterprise bean types, the remote and home interfaces of the enterprise beans, the primary key of each entity bean, and the dependent classes and interfaces. It also contains an XML deployment descriptor for the enterprise beans.

Resource Adapter aRchive (RAR) files are also Java archives that store XML files, Java classes, and other objects for J2EE Connector Architecture (JCA) connector.

Enterprise ARchive (EAR) files are platform-independent compressed Java archives that store the entire application structure for a Java enterprise application, consisting of WAR files, EJB JAR files, RAR files, and other JAR files as necessary. Multiple WAR, JAR, and RAR files may make up an EAR file. However, one EAR file represents one complete Java enterprise application.

Enterprise beans are Java components that typically implement the business logic of J2EE applications, as well as accessing data. The enterprise beans, packaged in EJB modules and installed in an application server, do not communicate directly with the server. Instead, the EJB container is an interface between EJB components and the application server. Together, the container and the server provide the enterprise bean runtime environment.

The container provides many low-level services, including threading and transaction support. From an administrative perspective, the container handles data access for the contained beans. A single container can host more than one EJB JAR file.

The way EJB applications are put together is described in greater detail in the *WebSphere Application Server ND 6.1 Information Centre*. This is a large HTML document describing how to develop, assemble, and deploy applications. It may be consulted at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/welc_content_cadm.html

12.2.5 Scalable service-oriented characteristics of the runtime

The WebSphere Application Server V6.1 product is designed to promote the flexible cross-network infrastructure needed for Service-Oriented Architectures (SOA).

Note: The software perspective of SOA

Service-Oriented Architecture (SOA) is an architectural approach that supports service orientation. Service orientation models the requirements of software users in the form of services. When SOA is used to compose business applications, then services are meant to convey repeatable business tasks that may be linked to meet the requirements of software users.

In an SOA environment, shared resources are made available as independent services that can be accessed without knowledge of their underlying implementation or programming technology. For example, services running in a J2EE runtime on Linux can be requested and consumed in the same application as services running in a runtime supporting the C# programming language in a different operating system. SOA helps users build composite applications, which are applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes.

SOA implementations may make use of different standards, such as the SOAP message format or EJB method calls and of different communication mechanisms, such as HTTP and Java Messaging Service. A key requirement is that the services may inter-operate according to a service definition contract, such as a Web Services Description Language (WSDL) document for each service.

Software architects can in principle implement an SOA solely using IBM WebSphere MQ, or using Common Object Request Broker Architecture, or using existing Remote Procedure Call (RPC) technology. If they are already working in a mainly J2EE development environment, they can leverage servlets and enterprise beans. Currently, Web services is becoming the most commonly adopted standard to support SOA. For information about the relationship

between Web services and SOA, or the application of IBM Patterns for e-business to a Web services project, refer to *Patterns: Service-Oriented Architecture and Web Services*, SG24-6303.

J2EE 1.4 provides full support for both clients of Web services and providers of Web services. It considers these to be endpoints that may be published, discovered, requested, and consumed by reference to a platform-neutral Web Services Description Language (WSDL). The relationship between the Web services endpoints is shown in Figure 12-5. If the services were created by Java developers, then the services to be made available as endpoints may be implemented as ways to invoke servlets, enterprise beans, or even, directly, Java classes. More generally, the specification allows different programming languages and technologies to participate as endpoints, which can be described using WSDL, so as to open up and encourage greater levels of enterprise software integration.

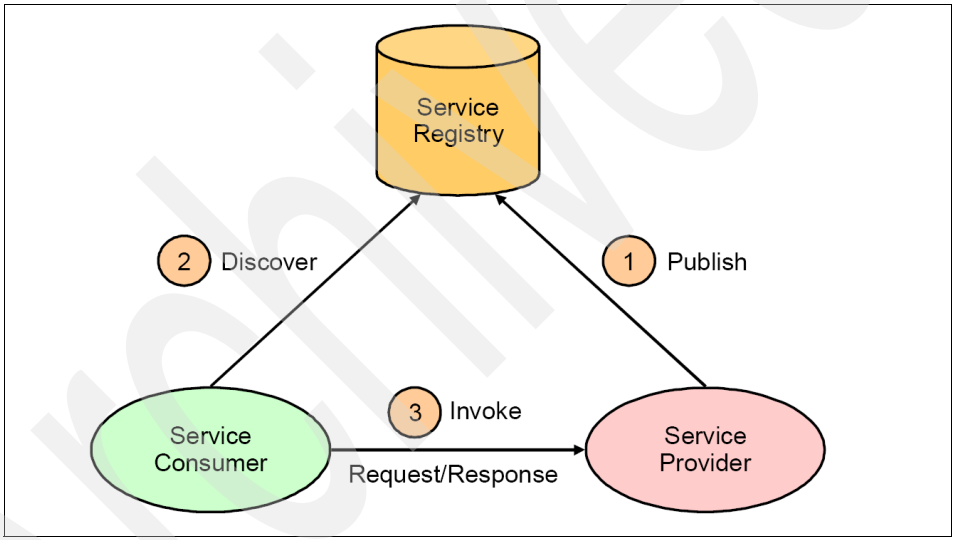


Figure 12-5 Relationship of service providers to service requestors to support SOA in J2EE 1.4

Several Java technologies work together to provide support for Web Services, as outlined in Table 12-1. The Web Services for J2EE specification fully defines the deployment of Web service clients and Web service endpoints in J2EE, along with the implementation of Web service endpoints using enterprise beans.

Table 12-1 Web services APIs provided by J2EE 1.4

API	Purpose
Java API for XML Processing (JAXP) 1.2	XML processing API, which supports different parser implementations along with validation and XML schema datatypes.
Java API for XML-based RPC (JAX-RPC) 1.1	API for building Web services endpoints and deploying them by using XML-based RPC as used for Simple Object Access Protocol (SOAP) 1.1 RPC calls.
Java APIs for XML Registries (JAXR) 1.0.4	API for accessing different kinds of XML registry servers, such as the Universal Description, Discovery, and Integration servers.
SOAP with Attachments API for Java™ (SAAJ) 1.2	API to produce and consume messages conforming to the SOAP 1.1 specification and SOAP with Attachments note.

At the time of writing, a comprehensive document covering Web services concepts and how to develop and deploy effective solutions in WebSphere Application Server is the book *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257.

The WebSphere Application Server V6.1 product has features that extend beyond the J2EE 1.4 based provisions for SOA with Web services.

The product runtime features:

- Extensive Web services support to allow interoperation between applications within an enterprise, across enterprises, or across the internet in a loosely coupled, language neutral, platform independent way. The Web services standards adopted cover the latest APIs to accomplish the following:
  - Message transmission and exchange of messages between definable endpoints.
  - Description and cataloguing of Web services so that they can be discovered and consumed.

- Reliable messaging so as to ensure that message exchanges are completed as part of a Web services solution.
- Transactional control over the sequence of operations completed in a Web services environment.
- Security enhancements to messaging APIs so that integrity, confidentiality, and overall security of services may be provided.
- Business Process Execution Language for Web services to provide a formal way to specify business processes and interaction protocols.
- Web services manageability, which is defined as a set of capabilities for discovering the existence, availability, health, performance, and usage, along with the control and configuration of each Web service.

These developing standards may be tracked at the Web site:

http://www-128.ibm.com/developerworks/webservices/standards/

- Particular Web services (WS) enhancements include:
  - WS-Notification, which enables Web Service applications to utilize the publish-subscribe messaging pattern.
  - WS-Interoperability Basic Security Profile, which detail requirements on the encrypted and unencrytped portions of messages to provide transport-neutral mechanisms to address Web services and to facilitate end-to-end addressing.
  - WS Business Activity, which provides protocols to help build applications that require consistent agreement on the outcome of long-running distributed activities.

12.2.6 WebSphere Application Server V6.1 product architecture

This Application Server provides installers with the ability to configure a resilient secure runtime. Parts of the application server functional runtime is shown in Figure 12-6 on page 411.

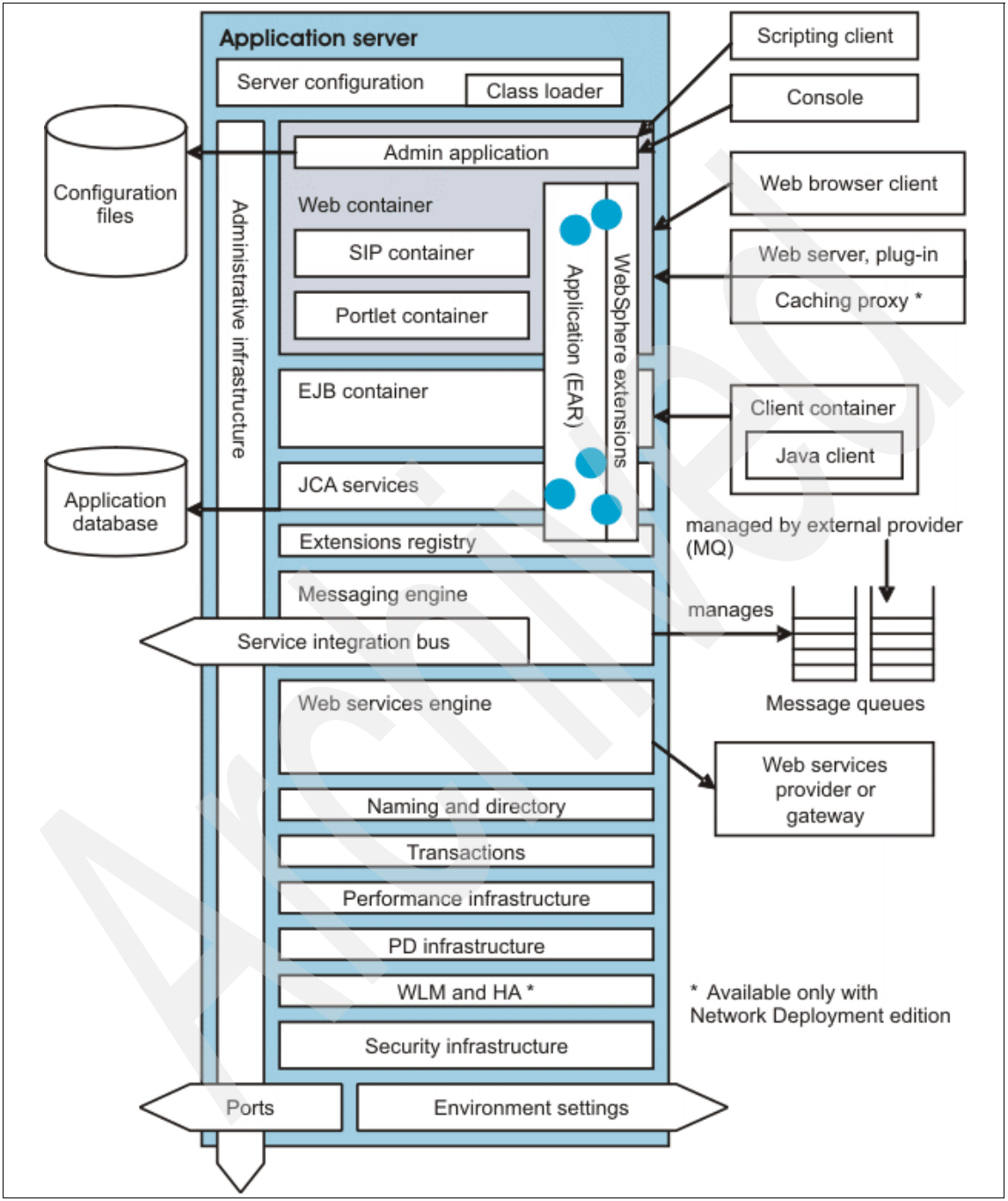


Figure 12-6 IBM WebSphere Application Server V6.1 product architecture

The application server functional runtime includes:

- A Web services engine to support the APIs discussed in 12.2.5, “Scalable service-oriented characteristics of the runtime” on page 407.
- A container of Session Initiation Protocol (SIP) servlets for developing instant messaging, voice, video, and other multimedia application services.

  A SIP container, following the JSR 116 API, is a server-side component which invokes the SIP action servlet and then interacts with the action servlet to process SIP requests. Application developers familiar with HTTP servlet programming may now combine HTTP servlets with SIP servlets, exported as SAR files, extending the reach of Java enterprise solutions.
- An embedded Portlet container for running JSR 168 compliant portlets.

  These portlets are user interaction fragments that may be assembled to form a completed portal page suitable to be served from a servlet or aggregated using JSPs or invoked from a remote server managed using the Network Deployment product.
- A Service Integration (SI) bus to support asynchronous messaging (JMS) and SOA applications.

  Each bus is configurable as link that can be started on each server machine or LPAR’s local messaging engine. Applications connect to a bus at one of the messaging engines associated with its bus members. The messaging engine manages bus resources and provides a connection point for applications. In its simplest form, a bus can be realized by a single messaging engine, one per LPAR. Any one messaging engine is uniquely identifiable and has its own unique store of messages. A SI bus can support either point-to-point or publish-subscribe asynchronous messaging schemes.

  After an application has connected to the bus, the bus behaves as a single logical entity and the connected application does not need to be aware of the bus topology. If the application uses the JMS API, then connecting to the bus and defining bus resources is handled by the administered JMS connection factory and JMS destination objects, under the JMS tab of the ISC.
- A transaction manager, whose purpose is to ensure the transactional behavior of submitted work units that may span requests or sequences of enterprise bean method invocations. For example, the work units may be inferred from the transaction settings provided in an EJB module that defines enterprise beans configured for container-managed transactions. In principle, the submitted work needs to be coordinated with a resource and the purpose of managing it is to ensure that each such resource is driven to a consistent outcome either at the end of a transaction or after a failure and restart of the application server. An example of such a resource is a database resource manager. Typically, the container of the enterprise application will manage this behavior transparently. In Linux systems, it has the ability to ensure single

phase commit across a number of JDBC resource managers from different vendors.

- A security infrastructure to supply information about the location, capabilities, and attributes, including user ID and password pairs and certificates, of resources and users known to each application. Each application server node can supply information for various security services (authentication and authorization). In addition to supplying the credentials, the security services are able to perform the actual security processing, for example, to verify certificates. This provides application-level security in addition to any "gatekeeper" security that may be provided at the HTTP server before the Java application begins to use resources to satisfy a request. Additional authentication and audit structures are available to the Java application itself, for example, to authenticate access to the enterprise tier.
- A distributed naming and directory service that implements the Java Naming and Directory Interface™ (JNDI) services. All containers within an application server require access to a common naming service. The deployment descriptor for each application component defines a particular set of name-value entries recorded in the shared JNDI. By this means, all application components are able to look up the values of particular entries as defined within the application or as supplied at deployment time.

There are extensions to the basic functional architecture to allow for greater scalability and increased availability provided specifically by the Network Deployment edition of the Application Server product.

12.3 The building blocks for WebSphere clustering

WebSphere Application Server ND provides a single point of administration of multiple nodes. It is possible to install Network Deployment on any machine in the network to create a network Deployment.

We have seen that a WebSphere Application Server may be administered by a web-fronted ISC or, alternatively, a wsadmin scripting language. In either case, we can extend this principle to administering a cluster of independent but functionally similar application servers.

Figure 12-7 is an example of a topology consisting of a deployment manager controlling the configurations of two nodes in the cell through their node agents, with one node running three application servers, and the other node running one application server.

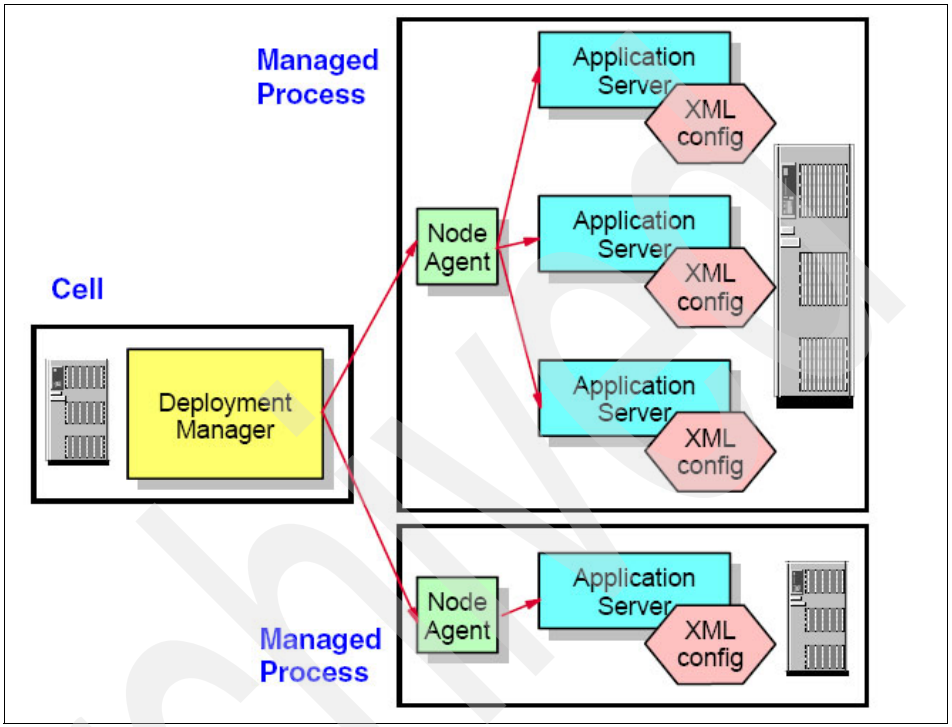


Figure 12-7 Deployment manager example

In order to o adminster a cluster, here are the underpinning concepts:

Managed server (process)
: Any JVM that can be managed in a WebSphere environment. The base example is a single application server. For a software cluster, we need application servers, messaging engines, and deployment managers. These are all launched as appropriately configured Java managed (servers) or processes.

Node agent
: These processes control the running of the application servers on the machine or logical partition and ensure that its XML configuration data are kept up to date as instructed by the deployment manager.

Cell A collection of managed processes that can be administered together as a single unit by one deployment manager. The cell should include all application servers and messaging engines that the deployment manager needs to be able to reach in the network.

Cluster A subset of the application servers and messaging engines that are selected from the cell. These members are all configured to serve one common Java application.

Profile A set of files that describe a single application server instance, defined on a common set of machine installation binaries (for example, the installation folders of Network Deployment 6.1.0.2). By default, these files are set up to configure a single application server. They are managed by a simple command-line tool that is capable of applying different templates. Try typing **/opt/IBM/WebSphere/AppServer/bin/manageProfiles.sh --help** to see the possibilities.

Master repository The centralized collection of all configuration XML files held on behalf of all managed servers in the cell. This collection needs to be periodically synchronized with the XML files held on each node, in their local file systems.

We shall base our description of performance of a real network implementing WebSphere Application Server ND on an example network. In our example, we shall place the deployment manager on a separate external machine and a distribution of the managed servers across available LPARs. We shall also assume the we have available some high capacity external machines to provide:

- The HTTP server tier at the client-facing front of network
- The DB2 database tier at the back end of the network

In our example, these have been tuned in line with the performance hints of the preceding chapters. So we shall concentrate on the performance of the ND cluster separately from the front- and back-end systems.

After we install and start the deployment manager instance, we can use the addNode utility to add each LPAR's WebSphere Application Server installation to the Network Deployment cell. For example, with our deployment manager running on machine, blade2ppc, at our first LPAR, w590lp1, we may invoke the following:

```
addNode.sh blade2ppc
```

Then we repeat this step at the second LPAR, w590lp2, and the third LPAR, w590lp3, so that we create altogether three nodes as members of the cell administered by the deployment manager for the network running on blade2ppc. The deployment manager assumes responsibility for the configuration of any application server nodes that belong to the cell.

Each managed server keeps all the data necessary to start itself. The configuration data is in XML files and the application data is in enterprise application EAR files. Using WebSphere Application Server ND, the deployment manager contains the master repository of XML files. Individual nodes and servers synchronize the files with the master configuration data, and all the changes applied to them directly are lost after synchronization.

The recommended way of working is to change configuration data only through the deployment manager. This may be achieved by attaching the adminstrative client to the deployment manager, which by default runs on port 9060, as indicated by our invocation of the ISC using the Web address in our lab:

```
http://blade2ppc:9060/admin
```

where the deployment manager is running on blade2ppc.

12.4 Virtualization and application server clustering

As we have seen in Chapter 3, “System p server virtualization” on page 49, virtualization gives us the opportunity to subdivide a physical server and its resources into multiple independent logical partitions all running their own system image. Each logical partition (LPAR) runs virtually as a server running its own Linux system image.

Virtualization promises to be useful to application server deployments because it provides increased topology choice for configuring multiple application servers. IBM WebSphere Application Server ND allows multiple servers to be grouped into a cluster so as to be ready to run the same application. Clustered applications can improve the reliability and availability of application components. All servers in the cluster can be made available to respond to user requests.

Application server clustering provides:

- Workload management between servers who are responding to Web address requests or RJB method calls
- Failover between servers who are responding to Web address requests or to EJB method calls.

Figure 12-8 shows two or more application servers administered by a deployment manager to achieve application server clustering.

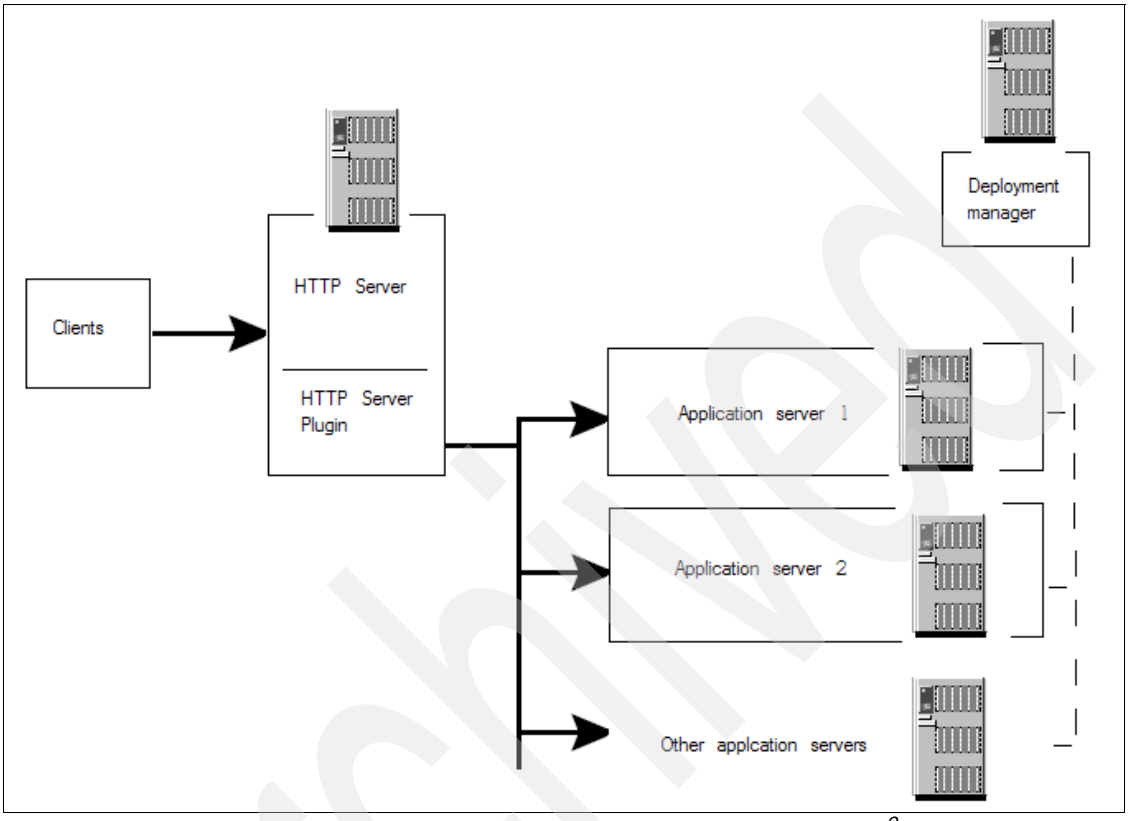


Figure 12-8 Application Server clustering from presentation to application logic tier[2]

The application servers shown in Figure 12-8 are members of the same cluster. Individual members of the same cluster are functionally identical copies of the configured applications, consisting of:

- EJB container configuration
- Enterprise beans defined
- Web container configuration
- Web modules and their contents, specifically servlets defined

[2] In this schematic, we abbreviate the deployment manager's link to each application server as a single dashed line. This is for brevity, so as to focus on the separate machines or LPARs we need in order to construct the cluster. The full connection from the deployment manage to a node agent which controls several application servers is shown in Figure 12-7 on page 414.

Each application server may reside on the same logical server or on different logical or physical servers, such as the three LPARs indicated in Figure 12-9, which shows the preparation of three LPARs allocating cores to host three nodes in an application server cluster administered by a deployment manager running on a separate machine.

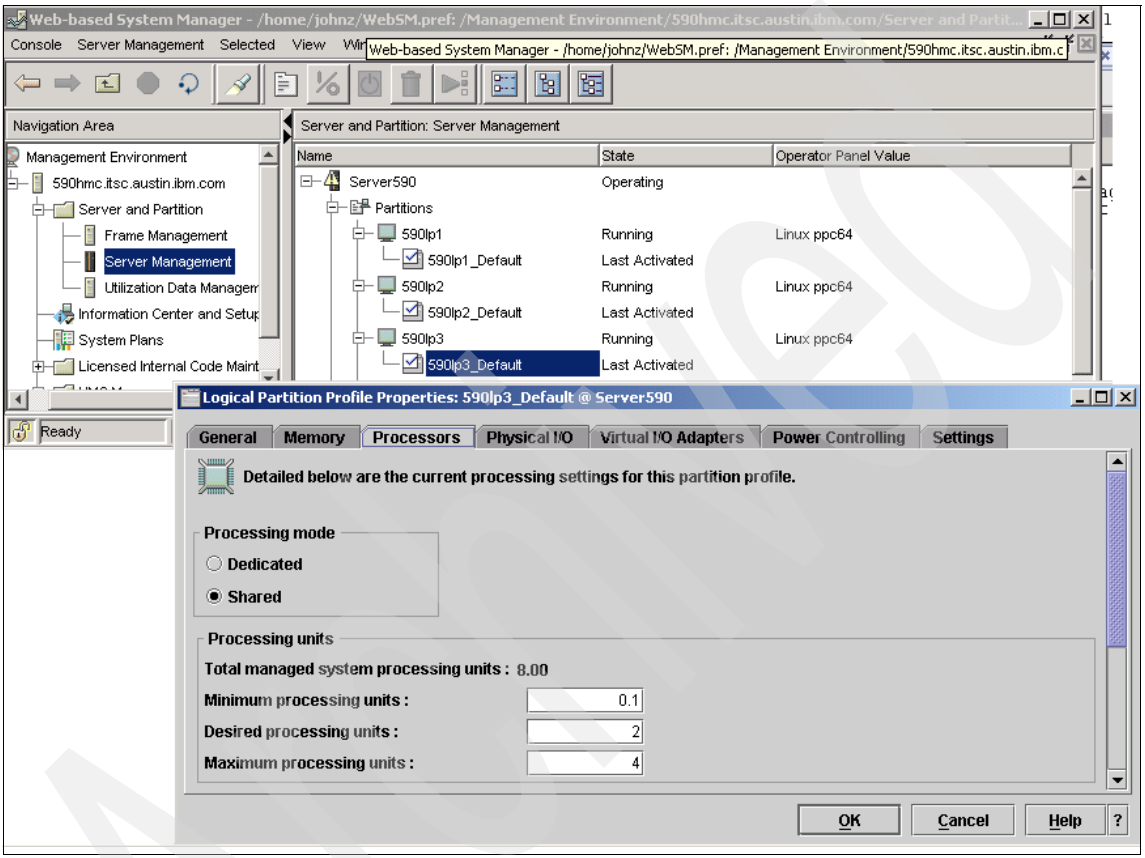


Figure 12-9 Preparing three LPARs

Assuming that we wish to scale an application across multiple (application server) JVMs, then there are two different options for scaling:

Vertical scaling The arrangement of application servers on a single logical or physical server machine. If there are multiple application servers providing the same application functions on the same server or LPAR, we have vertical scaling. This topology provides application server failover and load balancing across the cluster members.

Horizontal scaling The arrangement of application servers across multiple logical or physical server machines. If there is at least one application server available to run the application on each of our chosen servers or LPARs, then we have Horizontal scaling. The topology provides for application server failover and load balancing across the cluster members. There is additional control over availability because the run status of each server or LPAR is administered separately from the application server software.

Before horizontal scaling is possible, we must first ensure that each of the servers has been entered as a member of the cell administered by the deployment manager for the topology. To achieve this, we must add the server as a node to the running deployment manager, as shown in Figure 12-10. This is achieved by running the **addNode.sh** configurational instruction at the default application server profile of each of the servers where the application server code has been installed.

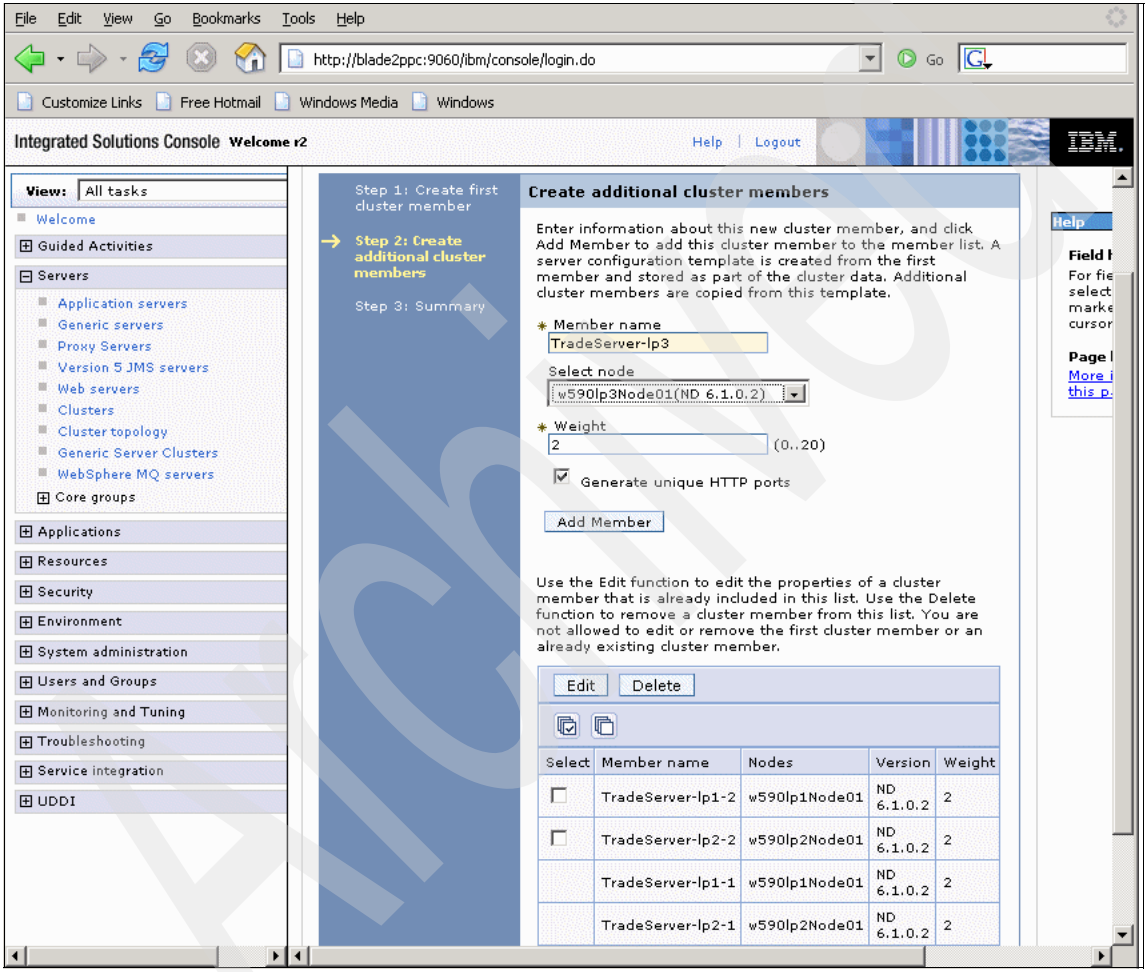


Figure 12-10 Creating a new cluster member on an existing node

After the nodes have been made available to the deployment, one per physical or logical server, then we may select the nodes we intend to include as members of the cluster. This may be achieved using the ISC or by scripting using **wsadmin**.

For example, in Figure 12-10 on page 420, we show the entering of a new node, based on an LPAR named w590lp3, into a cluster. The setup for the new node w590lp3Node01 provides a weight of two. The HTTP Server plug-in will be configured according to the weights of the individual nodes. It sends requests to alternate nodes in proportion to their weights. Ideally, the weights should be in proportion to the relative processing capacities of the individual nodes. In the situation where the nodes are similar physical machines or LPARs based on similar resources, then the weights should be the same for every node entered.

The topology for three equal server nodes running with our single TradeCluster is shown in Figure 12-11. It shows cluster topology using one appserver in each of the three nodes that are running in separate servers or LPARs.

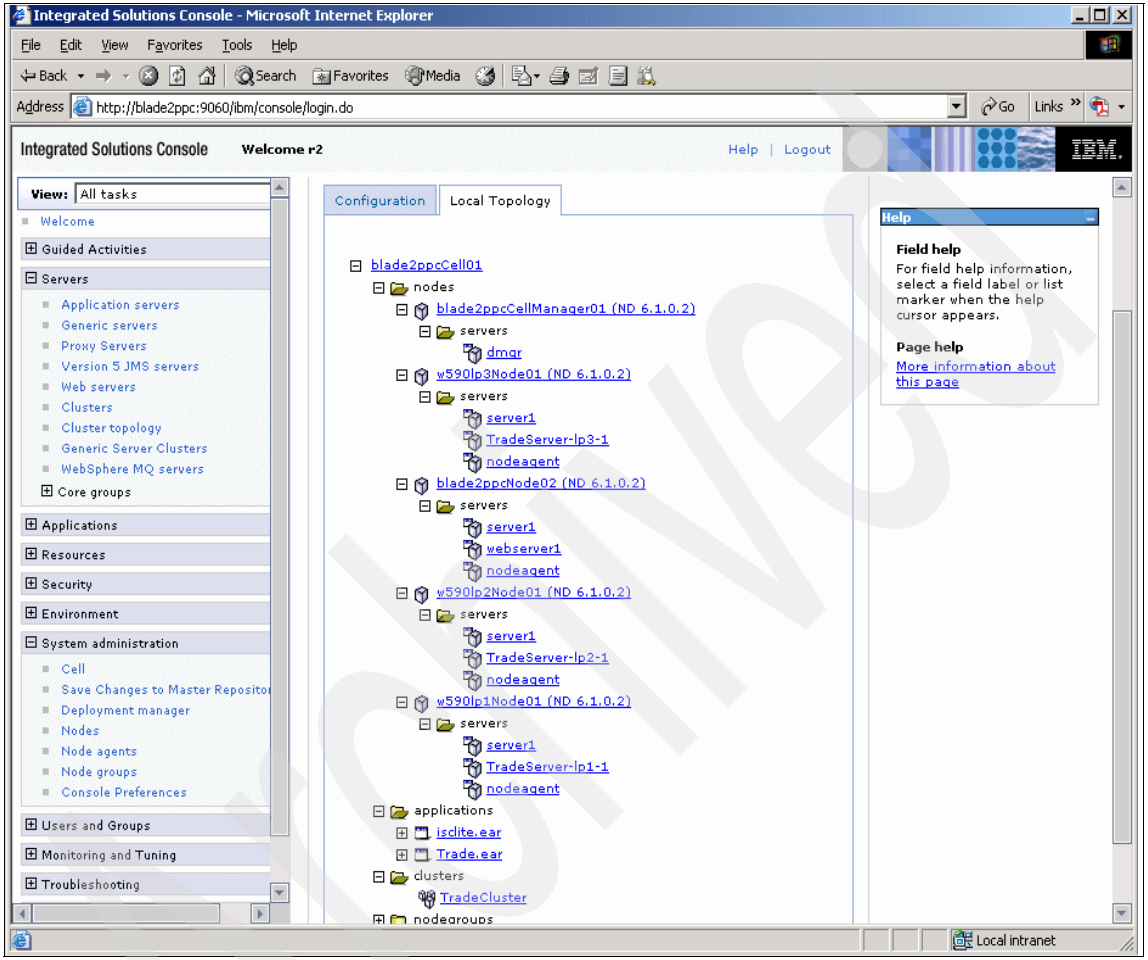


Figure 12-11 Cluster topology using multiple nodes

We can consider using the System p hardware at our disposal in a variety of ways. One option is to use a SMP machine as a single monolithic machine dedicated to a single application server. At the other extreme, we may partition the SMP machine into a set of smaller LPARS, each with the capacity to run multiple servers.

The machine setup for our test example topology is shown in Figure 12-12.

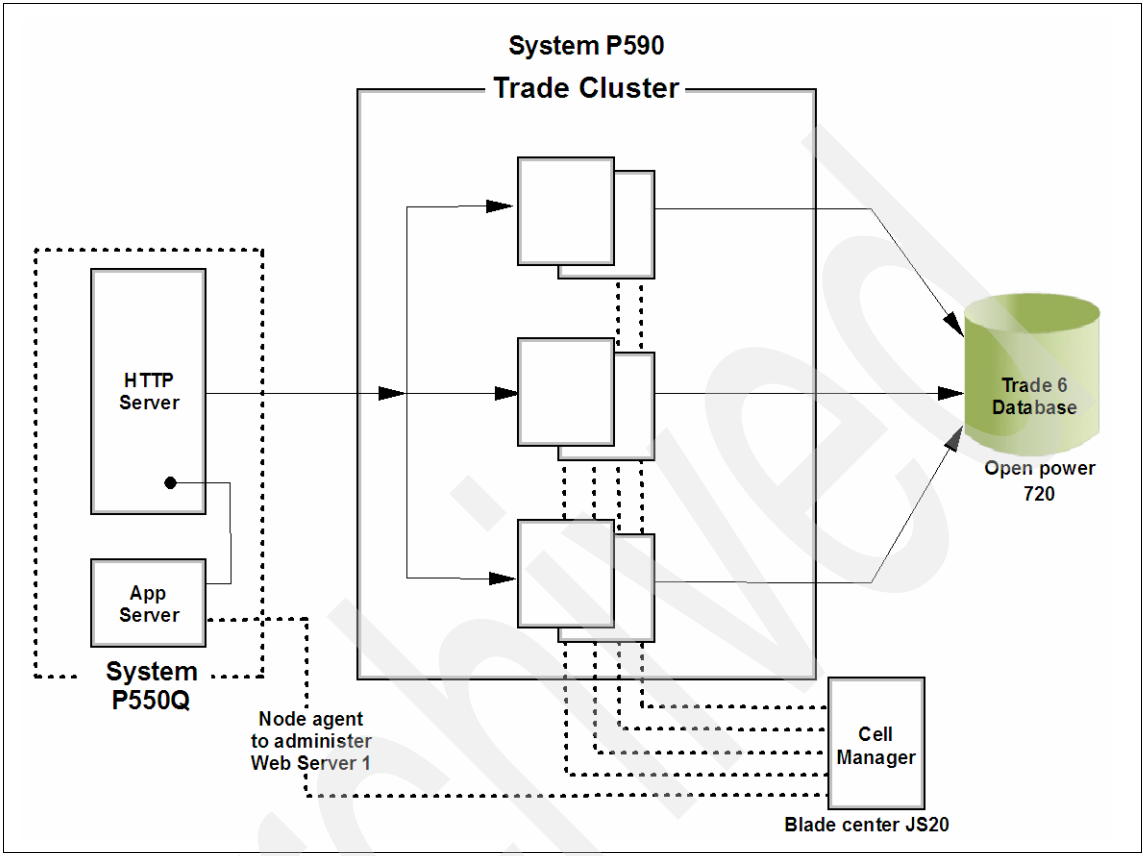


Figure 12-12 Topology of system under test

The topology of the system under test is shown in Figure 12-12 for evaluating scaling options, with a fixed external HTTP Server (on separate 550 LPAR) with WebSphere plug-in to handle routing and failover of HTTP requests a fixed back-end database on a dedicated 720 with external RAID storage array, and fixed deployment manager on blade2ppc node to control a number of application server cluster configurations on a 6-core 590 server.

We can vary the exact topology inside the dotted square labelled “application server cluster” by dividing up the System p 590 server in different ways.

We made several comparisons to the way a 6-core System p 590 server might be utilized through the Web-based system manager to configure the server. Here are some of the options we looked at:

- The “1 x 1” option: This is a single application server running in one LPAR consuming all six available cores. Although it is possible to configure this as a cluster with one member, it is functionally equivalent to a stand-alone single server solution.
- The “1 x 2” option: This consists of two application servers running in one LPAR consuming all six available cores. This is the simplest exploitation of vertical scaling.
- The “1 x 6” option: In this situation, we have scaled up to six application servers running in one LPAR consuming all six available cores. This is typically an overloaded use of processes for the number of cores and is not the usual practice.
- The “3 x 1” option: In this situation, we have created three LPARs that have capped allocations of two cores each. Each LPAR runs a single member of the application server cluster.
- The “3 x 2” option: In this situation, we have created three LPARs that have capped allocations of two cores each. Each LPAR runs two members of the application server cluster.

The complete Java Web site represents a large number of tuning variables, including the settings available at the front end (the HTTP Server) and the back end (the database). A fine overview of these is given in *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392. For the complete multi-tier architecture, we must remember that the application is implemented as a queuing system with requests queuing:

- At the Web server(s), subject to the number of server threads available
- At the Web container(s), subject to the size of the Web thread pool
- At the EJB container(s), subject to the size of the ORB thread pool
- At the data source(s), subject to the size of the JDBC connect pool

In the following sections, we review performance changes through clustering application server JVMs, connection pool changes, and comparing caching strategies.

For a more extended set of scaling performance options, which are general to any software platform, the reader is advised to consult *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392

12.5 Load testing using Trade 6.1

In order to make performance comparisons that are meaningful, we have adopted the Trade 6.1 testing scenario. Trade 6.1 is targeted at middleware environments based on J2EE 1.4 and driving database computations through relational databases, such as the current releases of DB2 and Oracle 10g.

The Trade test application is primarily used for performance research on a wide range of platforms and middleware components, as a representative test of user-to-business real-world workloads. User-to-business workloads are among the most demanding regarding transactional throughput and need to be set up with the expectation of high volumes. The high transaction count may swing considerably according to the state of bidding. Therefore, any trends in trading site performance are likely to be time sensitive.

Performance trends identified for the transactional workloads tested against the trade scenarios may not be typical of other enterprise applications, business models, or other real-world use cases. Some of the variables encountered may be illustrative, however. Our use of Trade 6.1 is to provide a workload that drives key performance components and features of the WebSphere Application Server V6.1 offering, together with associated middleware, to gauge the differences made by tuning at the application-software and operating-system environment parameter levels.

Using Trade 6.1 we can drive requests through a number of key WebSphere Application Server components including:

- The EJB 2.1 component architecture
- Message driven beans (MDBs)
- Web services (SOAP, WSDL, JAX-RPC, and enterprise Web services)

It permits detailed functional pre-tests to be made before any performance run, choosing an appropriate architecture as summarized in the splash window from the Trade application running in our cluster, as shown in Figure 12-13.

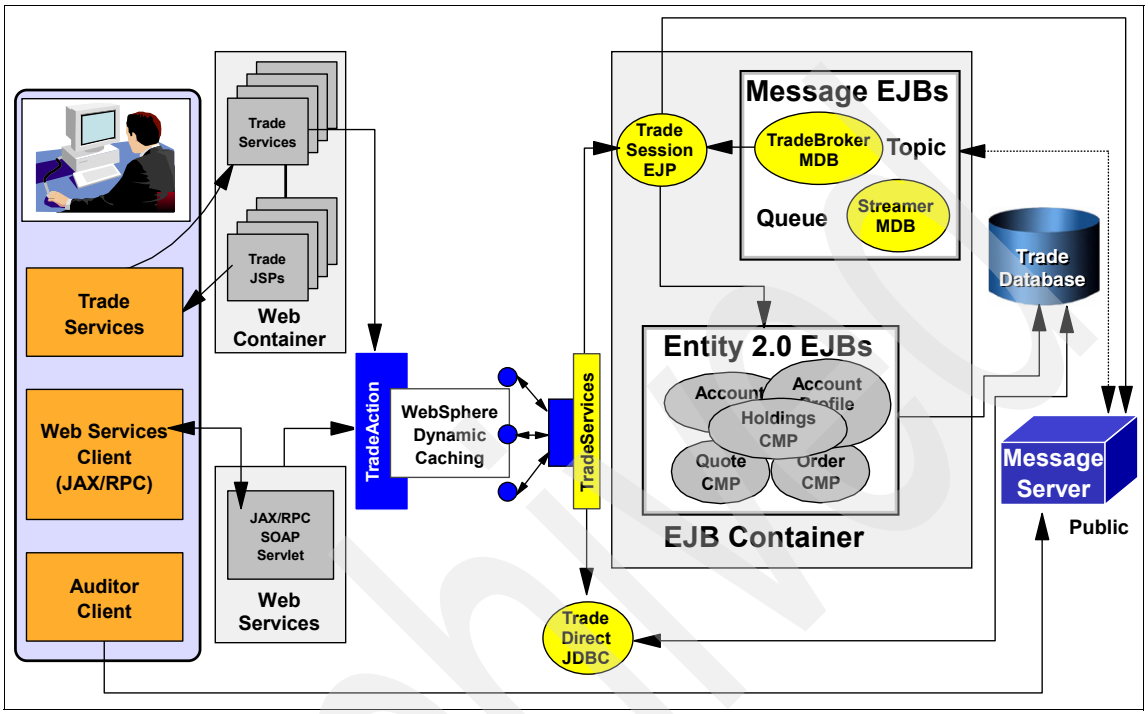


Figure 12-13 Trade 6 application running in a WebSphere Application Server ND V6.1 cluster

When we examined our EJB performance, we considered several runs of the EJB synchronous and EJB asynchronous (1-phase) with some different caching strategies. These are selectable from with the Trade 6 application's control user interface, as illustrated in Figure 12-14.

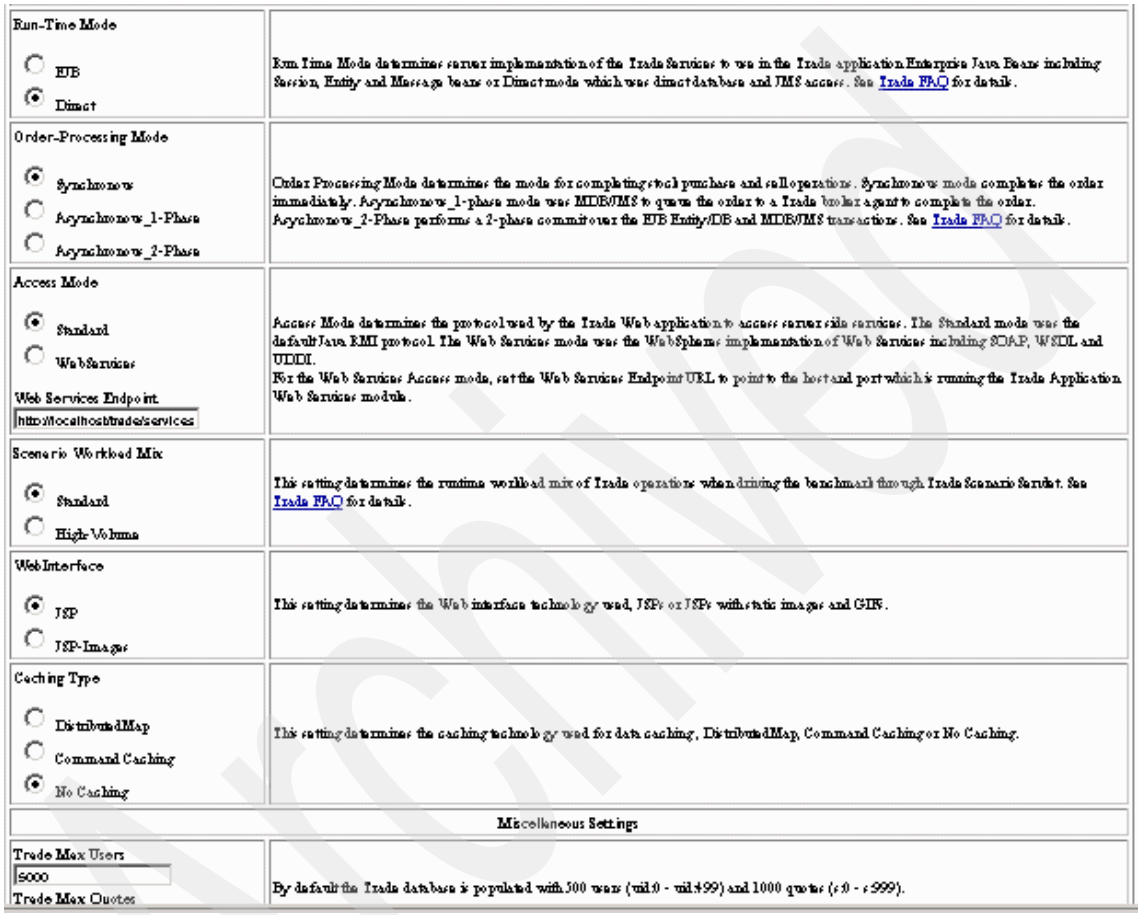


Figure 12-14 Trade 6 scenario options that are selectable for the cluster or per cluster member

Before starting a run, there are a number of initial tuning steps that are advisable.

12.5.1 First steps tuning system and application settings to remove performance bottlenecks

Before embarking upon any testing of the ability of application servers to allow the application to provide good levels of performance under concurrent load, there are some initial settings to review. These are the most basic settings to consider changing on System p SMP boxes for a broad class of different application workloads.

After the initial settings have been made, the tester needs to put in place procedures to continue to modify critical resource settings in the light of performance of the system under test. Suggestions for continuing tuning options follow in 12.5.2, “Peformance differences during software tuning” on page 435.

Server resource limits

The first of these is to check that resource availability limits for the Linux system are realistic on all the servers. Here is the definition of **ulimit**, which we established as appropriate across any of our System p servers for the user running the application servers and the cell manager, as shown in Example 12-1.

Example 12-1 Large ulimit settings where possible for application server nodes

```
blade2ppc:/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/logs/dmgr #
ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
pending signals                 (-i) 5888
max locked memory       (kbytes, -l) 32
max memory size         (kbytes, -m) unlimited
open files                      (-n) 99999
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 99999
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

It is especially important to provide a larger number of open file handles than the default.

Container settings

The next preparation step relates the server container settings. Each application server's ability to service Web requests is determined by the container settings. These may be updated in the online Integrated Solutions Console or through **wsadmin**.

For Web sites experiencing high volumes of Web requests per application server, a significant factor is how the Web container can hold a live session state in memory and how many pre-allocated threads it may keep. Both of these settings are resource dependent. We consider the situation of most application server nodes running on servers providing at least 2 GB or main memory, as illustrated in Example 12-2.

Example 12-2 Free memory on an application server node after configuration

```
[root@w590lp1 TradeServer-lp1-4]# free -m
      total       used       free     shared    buffers     cached
Mem: 4010        772       3238          0        181        268
-/+ buffers/cache: 322       3688
Swap: 1983          0       1983
```

In this situation it is appropriate to upgrade the session cache and thread pool as follows.

Session cache

For in-memory sessions (without database or in-memory replication of session objects), the size of the session cache is the number of sessions in the container's base session table. We can use the Allow overflow property to specify whether to limit sessions to this number for the entire session management facility or to allow additional sessions to be stored in secondary tables. When the session cache has reached its maximum size and a new session is requested, the session management facility removes the least recently used session from the cache to make room for the new one.

We set session cache size to reflect a realistic peak demand in the number of sessions we may experience, up to a ceiling of several thousands. We took our setting to 5,000 and we set the Allow overflow property to on. In our situation, this gave us some improvement in handling when the Web container was being highly stressed, as shown by its high CPU load (as shown by **vmstat**, **top**, or **nmon**, as preferred).

Web container thread pool setting

A thread pool enables components of the server to reuse threads, which eliminates the need to create new threads. Creating new threads expends time and resources.

For high peak loads, we typically need to set the maximum thread allocation higher. We doubled the minimum and maximum settings (to a minimum of 20 threads and a maximum of 100) and saw improvements in the number of client requests that our deployment could sustain.

An option that can be considered is to unrestrict the Web container thread pool so that it is allowed to overflow. There is a performance and security danger, however:

- In performance terms alone, an unlimited number of threads may present excessive stress to the application server, even if we control the number of clients to realistic numbers.
- If we do not have enough control over the reach and availability of the network, so we cannot put limits on peak client numbers, so we render our application vulnerable to denial of service attacks.

For these reasons, we need to consider which tier of the entire front end of the solution will act as the gatekeeper to clip peak swings in user demand. If there is no other available answer, the best decision may be not to allow the Web container threads to overflow. Although the Web application will most probably be higher performing by allowing the overflow, it is generally advisable to avoid the dangers by not checking the **Allow Overflow** box. As a compromise, it can be advisable to set the Web container thread limits to a value higher than the default.

Database connection pooling

At the front end of the application architecture, we may use thread pooling inside the Web container to control the volume of incoming requests. Assuming an application design that involves EJBs or JDBC, then while we increase the request volume, we are liable to find that the back end of the architecture starts to struggle to keep up with the increased number of database requests to be fulfilled.

To maximize the number of connections available, we need to have a number of pre-allocated SQL connections from the JDBC connection pool. We might decide to increase these if we know the number of agents to be created in the database can be correspondingly increased. In our earlier database chapters, Chapter 10, “Oracle Database” on page 299 and Chapter 11, “DB2 9 data server” on page 347, we saw that increasing the number of connections available to SQL statements could usefully be increased along with other database tuning

parameters. As we perform this database tuning, we often increase the size of the JDBC connection pool, above the default (of 10 connections). The connection pool size should always be less than the potential maximum number connections at the enterprise tier. If the enterprise tier is a DB2 database, then this is also called the maximum application count.

Be aware that in a clustered environment, the connection pool is a cell resource. This means that all nodes in the cluster share one pool. Adjust the pool in such a way that it is large enough to handle the requests of all the servers participating in the cluster.

Java Prepared Statement Cache

This is determined by the number of JDBC statements that are re-used with individual bindings throughout the lifetime of the application. Ideally, this can be determined by counting the number of new Prepared Statements created in the application code. Alternatively, the degree of re-use can be monitored and the cache adjusted if necessary.

In the case of the Trade 6.1 application, a JDBC Prepared Statement Cache size of 60 is advisable.

EJB container active pool and cache

The EJB container's inactive pool cleanup interval is a setting that determines how often unused EJBs are cleaned from memory. A high value will help ensure that instances that have already been encountered by the application on the given cluster member may be re-used. This prevents wasteful re-instantiation of the same entity bean.

Although it is often useful to increase the inactive pool cleanup interval, the penalty is the increased use of heap memory. For this reason, it is usually one of the parameters best tuned with the help of the Tivoli Performance Viewer.

A similar situation exists with the EJB container cache.

Tivoli Performance Monitor

A visual monitor of the running of application components is included with the IBM WebSphere Application Server product. The tool, the Tivoli Performance Monitor (TPM), is provided to show the current activity. For any application server, monitoring must first be switched on. The running system may be monitored once the Performance Monitoring Infrastructure (PMI) facility is enabled.

For a run of the application using the Trade 6 example, some trends in the run are shown on Tivoli Performance Monitor screen capture in Figure 12-15.

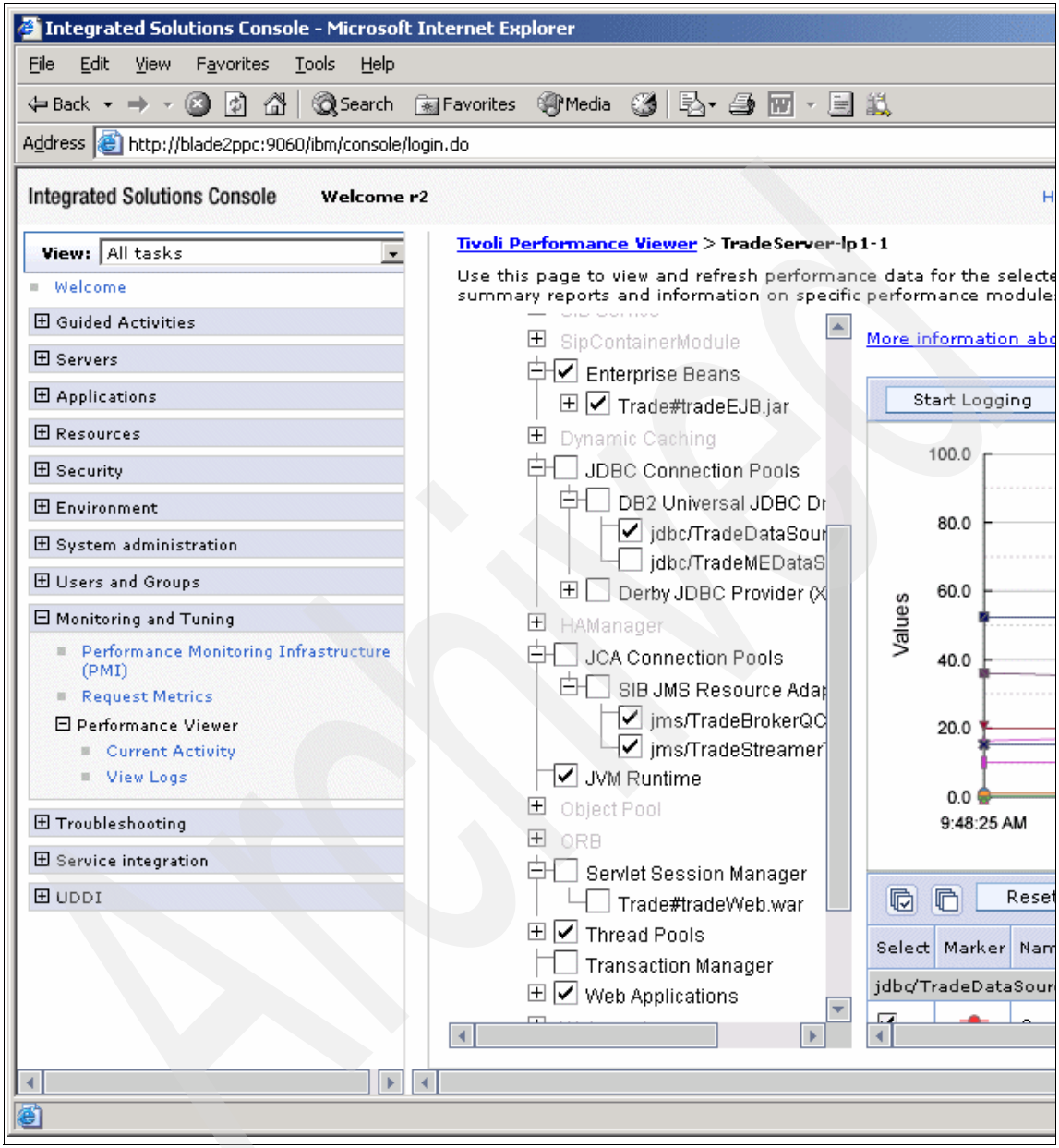


Figure 12-15 Tracing selected application server modules using Tivoli Performance Viewer

We need to scroll down to the definition of the trendlines, as shown in Figure 12-17 on page 436.

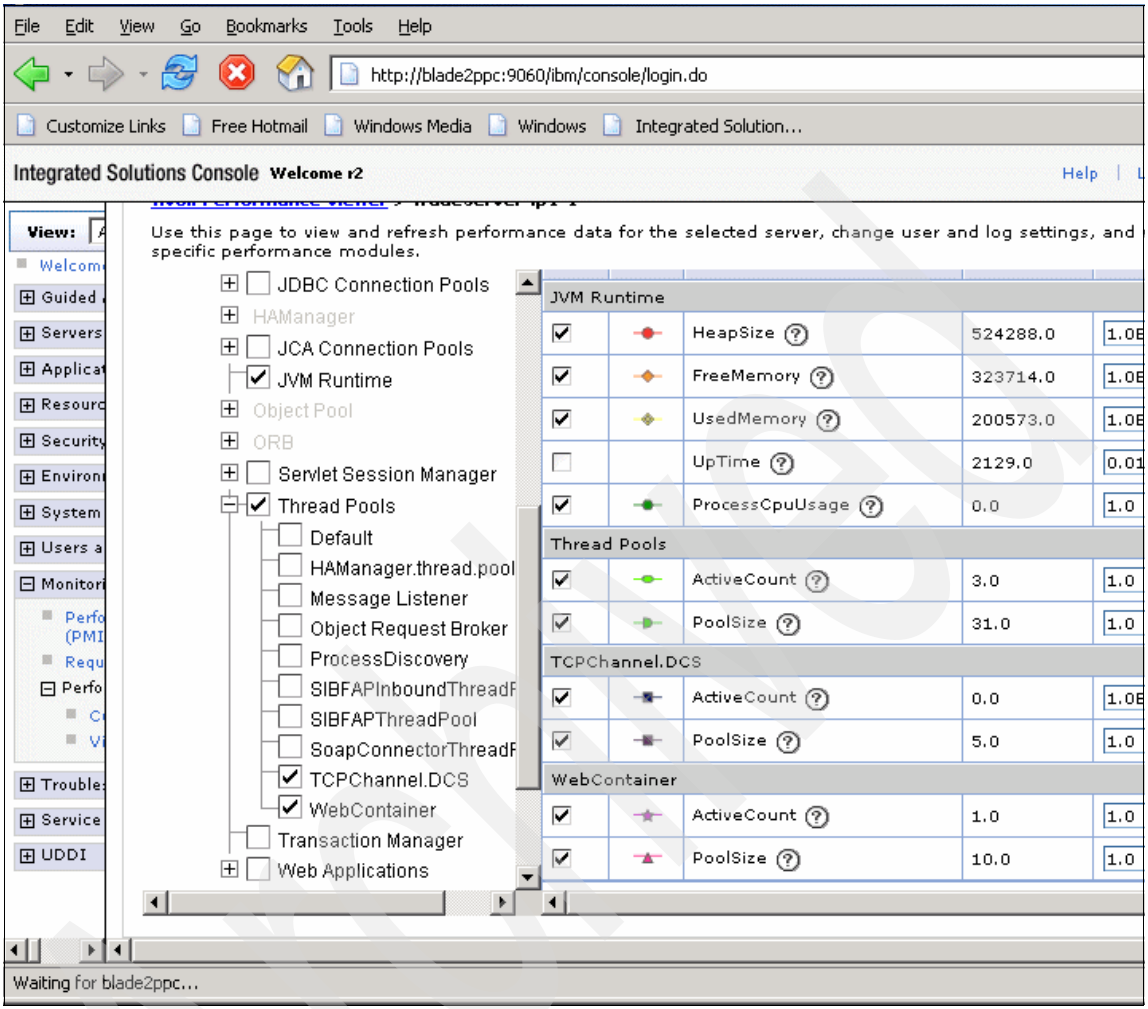


Figure 12-16 Key to lines displayed in previous figure

Running the TPV during the test runs of our application is the recommended approach in the areas of:

- EJB container tuning
- Messaging engine and SI bus tuning
- Security cache properties

The tuning options are analyzed in Chapter 6, “Identifying bottlenecks” on page 141.

Additionally, it provides a fine check on the preparatory settings we have made. For example, we can monitor JVM heap growth, as shown in Figure 12-16 on page 433, to determine that the GC frequency is stable and well contained within the available heap memory.

Heap size tuning

Maximum heap size determines how often garbage collection needs to run for the same number of Java object instantiations. We wish to avoid automatic memory management becoming so intrusive that space is reclaimed before enough useful processing gets done. Yet on the other hand, we wish to avoid the load of long, last garbage collections that may decrease the response time of methods that are waiting for an available thread. To get the balance right, we need to set up a number of short runs under representative test conditions.

If possible, tune the JVM heap by running the application in test mode, under simulated load with verbose garbage collection turned on, using the -verbosegc JVM argument. Then, tune the JVM heap to reach a garbage collection (GC) cycle rate of 5% to 20% of the total running time. In 64-bit JVMs, it is known that this can result in a significant memory footprint increase over the default.

A possible side effect of the memory footprint increase is a decrease in the L1 cache efficiency of the processor. This can be compensated by the L2 and L3 cache efficiencies of current POWER5+ processors.

Our test scenarios to make this assessment were 10 minutes long. We found that with a maximum and initial heap size setting of 512 MB, then the frequency of GCs in a single server was at around 10% of the run interval (1 minute), by inspection of the timings given in SystemOut.log. In our test runs, slightly increasing the heap size settings had no significant effect for the period we made this observation (less than one day in our case). Although reducing the heap size tended to produce more frequent GCs, we preferred the setting of about one half GB as being more realistic for the 590 server machine in production. We also studied the LPAR's overall memory trends (consulting a graph of **nmon** 5 second snapshots) during the run and found that, in every run, memory consumption increased just once and then stayed level (at a plateau) for the duration of the run using this setting.

12.5.2 Peformance differences during software tuning

With important bottlenecks already avoided, we may attempt some near realistic performance comparisons with respect to our example cluster arrangement of Figure 12-11 on page 422.

Using the Trade 6 test scripts, we are in a position to drive various workloads through our candidate cluster topologies. We use a workload simulator tool to control the generation of a number of virtual users. These are simulated users with wait times and inter-start-up delays to allow a load of many thousands to be progressively achieved. It must be remembered that despite the semi-randomized waits in the scripts, they are extremely demanding in comparison to realistic human users, because they are never more than a fraction of a second from entering their next input. Real humans have considerably longer waits and pauses than we have modelled in our example and the tester might consider sampling a large number of real users for a more realistic scenario.

We were able to run our test scripts on a bank of eight Linux blade servers, orchestrated from a controlling server. We determined that this arrangement had enough processing capacity to create thousands of virtual clients (implemented as Java threads in the load injection engines) using Trade 6 test scenarios involving EJB application design.

In Figure 12-17, we show a typical runtime trend line of how the virtual user load was stepped up. We use the cases where response times through the run did not exceed 1 second for more than 5% of the run. We examined runs lasting for exactly 10 minutes.

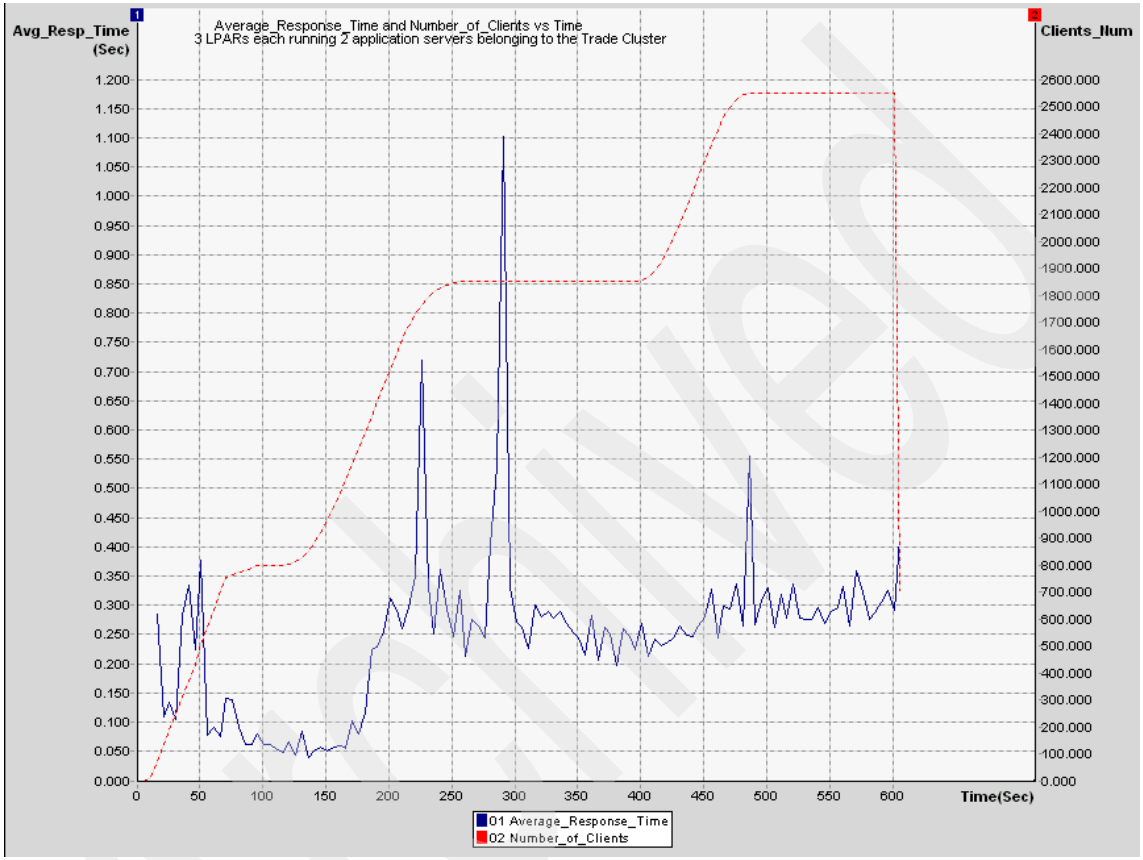


Figure 12-17 Average response time (left side axis) and number of clients (right side axis) under increasing load over the duration of a test run lasting 10 minutes

Measurements

The goal of performance tuning is to maximize the throughput for an acceptable response time.

Throughput The number of transactions that can be satisfied concurrently during a given time interval.

We measured the number of requests relative to our sampling rate of five seconds. If we multiply the observed readings by five, then it might be possible to transform these results into transaction per second counts, because this figure is often quoted as a statistic.

Response time The total round-trip time taken from the moment the client presents a request to the browser (as simulated by the virtual user) to the moment the result of the HTML page returns to the browser.

For our purposes, we assumed that the workload was being responded to quickly enough, provided the response time was below the sub-second level.

We analyzed our metrics as shown in the following sections.

Monitoring

All runs were preceded by two minutes of warm up testing. This should be effective in ensuring that JSPs have compiled and bean caches have been populated as far as realistic production sites might expect.

There were some housekeeping steps between each run:

1. The TRADE6 remote DB2 database was zeroed (all rows removed) and made ready to be repopulated.
2. The initial database was repopulated with 5000 user data, each having 2000 available stock quotes.
3. A new run of the **nmon** speadsheet tool was kicked off to conincide with the first requests from the load generator, using **nmon -F nameOfRun -s 5 -c 132**, where *nameOfRun* was a description of the test for later harvesting of the **nmon** information after the run had been completed, with snapshots taken every five seconds for a run lasting exactly 11 minutes (giving 132 snapshots).
4. **nmon** was used locally at the HTTP Server and DB2 database server machines to verify that these machines were extremely lightly loaded, to the extent of being oversized, as discussed in our assumptions.

After some initial experimentation, we found that it is possible to drive loads into the test up to 2400 virtual users before extreme overload was experienced within the containers. Driving up to the limit, the various tuning changes discussed in this section were attempted while using Tivoli Performance Viewer to monitor individual application servers. TPM was then left with monitoring turned off to collect the measurements from the load generation engines.

Comparison

In our test runs, we used the peak throughput, measured as the greatest total number of transactions achieved during a five second interval, as the measurement we would compare.

We have shown the results we obtained in Figure 12-18.

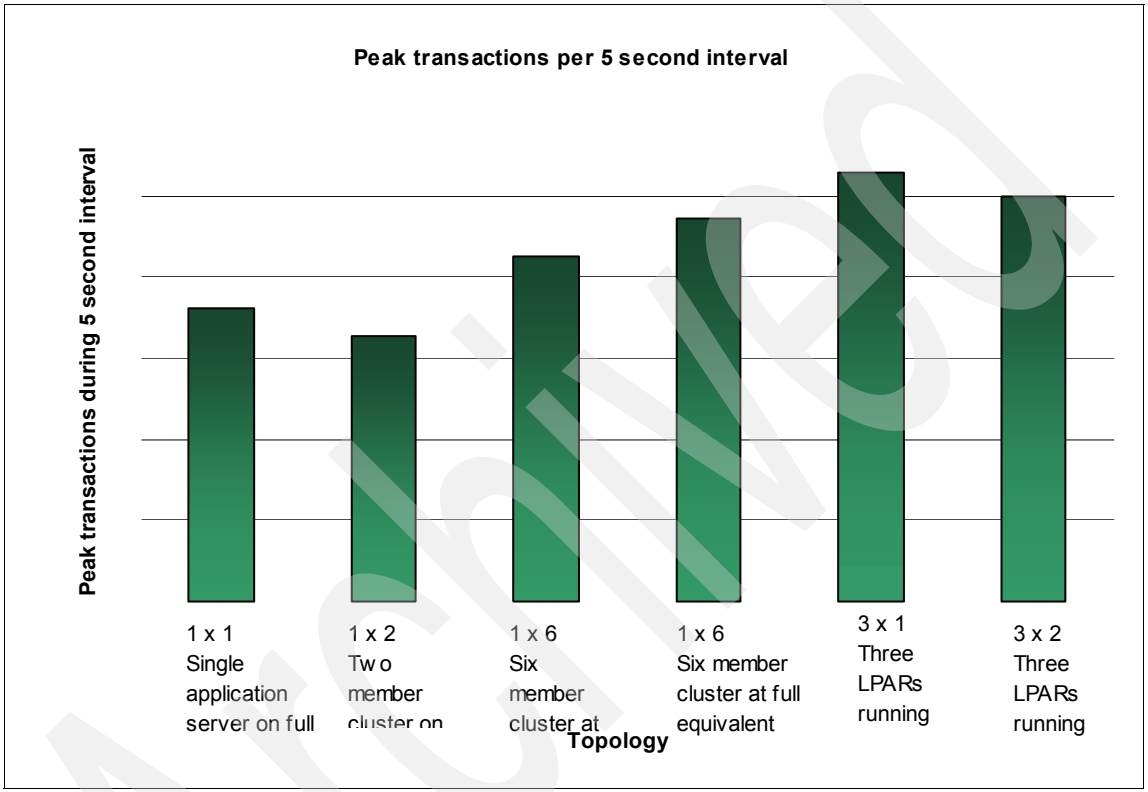


Figure 12-18 Peak transactions observed using synchronous method calls with application servers located in different cluster topologies

The results indicated that vertical scaling was not helpful to the peak throughput figure until exactly six application servers were added. For our application scenario using Web containers and EJB containers serving employing enterprise bean calls, horizontal scaling was quite effective.

12.6 Memory tuning and utilization effects

In the preceding sections, we have investigated the performance of an application server cluster using ordinary Linux system settings to remove basic WebSphere bottlenecks.

We now review the performance impact of two capabilities of Linux system tuning appropriate to SMP server nodes. These changes are:

- Large page sizes
- Caching service changes

12.6.1 Large page sizes

We described the new 16 MB page sizes provided with POWER 5+ in “Large memory page” on page 328.

We now consider the effect of setting the number of huge pages available to 100. The objective of this change allows the application server to access a JVM heap of 1.6 GB (=100 x 16 MB) in large pages. The setting is made permanent by altering the value stored in the file /proc/sys/vm/vm.nr_hugepages, as shown in Example 12-3.

Example 12-3 Specifying 100 large pages to the server node

```
[root@w590lp1 TradeServer-lp1-1]# free -m
             total       used       free     shared    buffers
cached
Mem:          7513        414       7099          0         15
154
-/+ buffers/cache:        244       7269
Swap:         1983          0       1983
[root@w590lp1 TradeServer-lp1-1]# sysctl -w vm.nr_hugepages=100
vm.nr_hugepages = 100
[root@w590lp1 TradeServer-lp1-1]# free -m
             total       used       free     shared    buffers
cached
Mem:          7513       2014       5499          0         15
154
-/+ buffers/cache:       1844       5669
Swap:         1983          0       1983
[root@w590lp1 TradeServer-lp1-1]# grep -i Huge /proc/meminfo
HugePages_Total:   100
HugePages_Free:    100
Hugepagesize:    16384 kB
```

Additionally, we must change the application server's JVM setting: select the **Server Infrastructure** tab on the ISC screen shown in Figure 12-15 on page 432 and then select **Process Definition** from Java and Process Management. Inside the box labelled Generic arguments for the Java Virtual Machine settings, we add the -Xlp option before relaunching the application server.

Additionally, where the server node is used to run just one application server and the large pages are entirely dedicated for it to use, then in this example, we set the maximum heap size to 1600 MB. This is the total available large page size memory.

Because the heap size is made up of large page memory and this is allocated as system-wide shared memory, we should adjust the value of **shmmax** to the sum of approximately 1600 MB + 100 MB added on, as shown in Example 12-4.

Example 12-4 Shared segment size calculation and setting

```
[root@w590lp1 kernel]# echo \
> $(((100 * 16 * 1024 * 1024) + (100 * 1 * 1024 * 1024)))
1782579200
[root@w590lp1 kernel]# sysctl -w kernel.shmmax=1782579200
kernel.shmmax=1782579200
```

After the relaunch of the affected server (TradeServer-lp1-1 in our example), we should see a process running on the server using the large page setting. A server running this process is shown in Example 12-5.

Example 12-5 Application server process using large page memory setting

```
[root@w590lp1 TradeServer-lp1-1]# ps -ef | grep java | grep Xlp
root     13570 12108 23 13:03 pts/0    00:01:06
/opt/IBM/WebSphere/AppServer/java/bin/java -Declipse.security
-Dwas.status.socket=34331
-Dosgi.install.area=/opt/IBM/WebSphere/AppServer
-Dosgi.configuration.area=/opt/IBM/WebSphere/AppServer/profiles/AppSrv0
1/configuration -Djava.awt.headless=true
-Dosgi.framework.extensions=com.ibm.cds
-Xshareclasses:name=webspherev61_%g,groupAccess,nonFatal -Xscmx50M
-Xbootclasspath/p:/opt/IBM/WebSphere/AppServer/java/jre/lib/ext/ibmorb.
jar:/opt/IBM/WebSphere/AppServer/java/jre/lib/ext/ibmext.jar -classpath
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/properties:/opt/IBM/WebS
phere/AppServer/properties:/opt/IBM/WebSphere/AppServer/lib/startup.jar
:/opt/IBM/WebSphere/AppServer/lib/bootstrap.jar:/opt/IBM/WebSphere/AppS
erver/lib/j2ee.jar:/opt/IBM/WebSphere/AppServer/lib/lmproxy.jar:/opt/IB
M/WebSphere/AppServer/lib/urlprotocols.jar:/opt/IBM/WebSphere/AppServer
/deploytool/itp/batchboot.jar:/opt/IBM/WebSphere/AppServer/deploytool/i
```

```
tp/batch2.jar:/opt/IBM/WebSphere/AppServer/java/lib/tools.jar
-Dibm.websphere.internalClassAccessMode=allow -Xms256m -Xmx1600m
-Dws.ext.dirs=/opt/IBM/WebSphere/AppServer/java/lib:/opt/IBM/WebSphere/
AppServer/profiles/AppSrv01/classes:/opt/IBM/WebSphere/AppServer/classe
s:/opt/IBM/WebSphere/AppServer/lib:/opt/IBM/WebSphere/AppServer/install
edChannels:/opt/IBM/WebSphere/AppServer/lib/ext:/opt/IBM/WebSphere/AppS
erver/web/help:/opt/IBM/WebSphere/AppServer/deploytool/itp/plugins/com.
ibm.etools.ejbdeploy/runtime
-Dderby.system.home=/opt/IBM/WebSphere/AppServer/derby
-Dcom.ibm.itp.location=/opt/IBM/WebSphere/AppServer/bin
-Djava.util.logging.configureByServer=true
-Duser.install.root=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01
-Djavax.management.builder.initial=com.ibm.ws.management.PlatformMBeanS
erverBuilder -Dwas.install.root=/opt/IBM/WebSphere/AppServer
-Dpython.cachedir=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/temp/c
achedir -Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
-Dserver.root=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01 -Xlp
-Djava.security.auth.login.config=/opt/IBM/WebSphere/AppServer/profiles
/AppSrv01/properties/wsjaas.conf
-Djava.security.policy=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/p
roperties/server.policy com.ibm.wsspi.bootstrap.WSPreLauncher -nosplash
-application com.ibm.ws.bootstrap.WSLauncher
com.ibm.ws.runtime.WsServer
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config blade2ppcCell01
w590lp1Node01
TradeServer-lp1-1
```

Before we set our use of large page size memory to 1.6 GB, we must check that there is enough total memory for all the jobs that the Linux system may need to run. There is a need to provide extra physical memory for the operating system and for at least one nodeagent JVM if we are running in a clustered environment.

In order to be able to access the server machine, we needed to ensure that our large page example was running on an LPAR with more than 2 GB allocated to it. We chose a 8 GB partition to implement this example.

Note: Precaution before setting up large page memory

If using large pages for better heap performance of a WebSphere application server instance, be sure to establish that there is sufficient total memory on the server machine. Failure to provide enough total memory can result in significant paging when running the operating system alone, even before other jobs are scheduled.

An important architectural design performance optimization for Java enterprise applications running in the WebSphere Application Server environment is to use dynamic caching.

Dynamic caching The caching of runtime execution results.

Significant dynamic caching may be performed at the application logic tier. We may contrast this with other caching that occurs at or before the presentation client-facing tier, as supplied by the HTTP Server. There is additional caching that can occur in the enterprise tiers and other points of entry into the total application solution network. However, for tuning purposes, our attention is focused on dynamic caching internally supported at the presentation and application logic tiers. This form of caching is strongly supported by efficient memory retrieval organization. So, large page memory configuration is likely to have the greatest impact on applications that make use of dynamic caching. We explore this area in the next section.

12.6.2 Dynamic caching services

In the IBM WebSphere Application Server V6.1 product, we can make use of the dynamic cache service to improve application performance by caching the output of servlets, commands, Web services, and JSPs.

The dynamic cache service works within an application server JVM by intercepting calls to cacheable objects. For example, it intercepts calls through a servlet service method or a command execute method and either stores the output of the object to the cache or serves the content of the object from the dynamic cache. The cache settings are held in the file cachespec.xml, which is part of the Web container definition (WAR file). So once the cacheable elements have been defined for the application, these settings are part of the deployment of the application onto all the nodes of the cluster.

There are two types of cache instance: one for servlets and JSPs and the other for objects.

Servlet cache instance
This is a memory location in addition to the default dynamic cache, where dynamic cache can store, distribute, and share the output and the side effects of an invoked servlet. In addition to being a memory location, it may optionally also be written to disk.

Object cache instance
This is a memory location in addition to the default shared dynamic cache, where J2EE applications can store, distribute, and share objects. In addition to being a memory location, it may optionally also be written to disk.

These cache instances hold servlets or objects that can be retrieved by JNDI lookup across all the application servers in the cluster, taking advantage of data replication services. The JNDI name that is specified for the cache instance in the administrative console maps to the cache-instance element in the cachespec.xml configuration file. Any cache-entry elements that are specified within a cache-instance element are created in that specific cache instance. Any cache-entry elements that are specified outside of a cache-instance element are stored in the default dynamic cache instance.

The cache-entries allow for invalidations where the cache value is emptied. Invalidations are transmitted across the cluster to keep cache values consistent and valid.

The object cache can be configured so that objects are recoverable by being flushed to disk when the application server is stopped and reloaded when the application server restarts. Each individual cache-entry can be stipulated as pushed to all servers, and pulled when needed by name or non-stared.

To exploit available caching settings within the Java application code, WebSphere Application Server V6.1 recommends using the API supplied by com.ibm.websphere.cache.DistributedMap.

Using DistributedMap API calls, J2EE applications can cache and share Java objects by storing a reference to the object in the cache. Whenever there is a hit to the cache, then CPU, memory, and network traffic are avoided, because the result is directly retrieved rather than computed by the application component running in the Web container or EJB container.

For more information about the DistributedMap interfaces, please consult the WebSphere Application Server product documentation at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tdyn_distmap.html

Dynamic caching is complex because it requires an understanding in the object design of the application. Dynamic caching is a suitable option for application designers to specify the “liveness” of objects determined by the length of time for which they hold unchanging values. Explicit timeouts can be set up to be sure that stale values are not held in the cache. In addition, by stating dependencies on other objects, the designer allows values to become invalidated based on programmatically evaluated runtime conditions.

In a clustered environment, the content you place in cache might be shared with other servers in the cluster. The content might also be offloaded to disk (provided that the cached objects have been defined as serializable).

Trade is designed as a performance sample application for the WebSphere dynamic caching service. By default, the caching capabilities within Trade are disabled. However, the caching type can be modified during runtime to use the DistributedMap interface.

The Trade 6.1 application defines various servlets and trade command objects that can be configured for caching using a suitable caching strategy. In WebSphere Application Server V6.1, the recommended caching strategy is DistributedMap.

The command objects defined in Table 12-2 are available for being held in cache when selected from the Trade 6.1 application by taking the DistributedMap option using the scenario selection page shown in Figure 12-14 on page 427.

Table 12-2 Trade 6.1 commands that have cache entry definitions in cachespec.xml.

Command definition	Lifetime of cache entry
MarketSummaryCommand	Invalidated on a timeout.
QuoteCommand	Quote symbols and prices. Invalidated when symbol is updated.
UpdateQuotePriceCommand	Update price for stock symbol. Invalidates QuoteCommand cache entry.
AccountCommand	User's account information. Invalidated during user trading.
AccountProfileCommand	User's account profile. Invalidated when profile is updated.
HoldingsCommand	User's stock holding. Invalidated when user's orders complete.
OrdersCommand	User's orders. Invalidated when a new order is entered to be bought or sold.
OrderCompletedCommand	User has bought or sold. Invalidates AccountCommand, HoldingsCommand, and OrdersCommand.
BuyCommand	New order created. Invalidates OrdersCommand.
SellCommand	New order created. Invalidates OrdersCommand.
LoginCommand	User's account. Invalidated if user logs out.

Command definition	Lifetime of cache entry
LogoutCommand	User's account. Invalidated if user logs in.
UpdateAccountProfileCommand	Update user's account. Invalidates AccountProfileCommand.
ClosedOrdersCommand	Cached for user and invalidated when an order is completed.

Also, the servlets and JSPs shown in Table 12-3 may be held in the servlet cache instance on each cluster member or application server.

Table 12-3 Trade 6.1servlets and JSPs that have cache entry definitions in cachespec.xml

Servlet URI	Lifetime of cache entry
/app	Sample portfolio application.
/register.jsp	For entering new users.
/marketSummary.jsp	Invalidated when database is reset.
/displayQuote.jsp	Displaying the quoted price until this is invalidated.

The commands shown in Table 12-2 on page 444 are non-shared. This option is useful where there is session affinity (in the case of a Web client, or transaction affinity in the case of a plain EJB client).

To enable the dynamic caching service, using the ISC or **`wsadmin`**, we take the following steps:

1. At each application server, navigate to Container services and select **Dynamic cache service**.
2. Navigate to the Web container and enable the servlet caching by clicking the **Enable servlet caching** radio button under the Configuration tab.
3. Configure a cache entry for elements we can cache by editing the file /opt/IBM/WebSphere/AppServer/profiles/AppServ01/properties/cachespec.xml.[3] In our case, we choose a version of the Trade 6 test application that has prepared the entry point references.

[3] This path assumes that we are using the default application server profile from an installation of IBM WebSphere Application Server ND. If the installation has been based on a default installation of WebSphere Application Server base, then the path opt/IBM/WebSphere/AppServer/profiles/default may be substituted for opt/IBM/WebSphere/AppServer/profiles/AppServ01 throughout this section.

In our test runs, we used the peak throughput, measured as the greatest total number of transactions achieved during a five second interval, as the measurement we would compare.

We have shown the results we obtained in Figure 12-19.

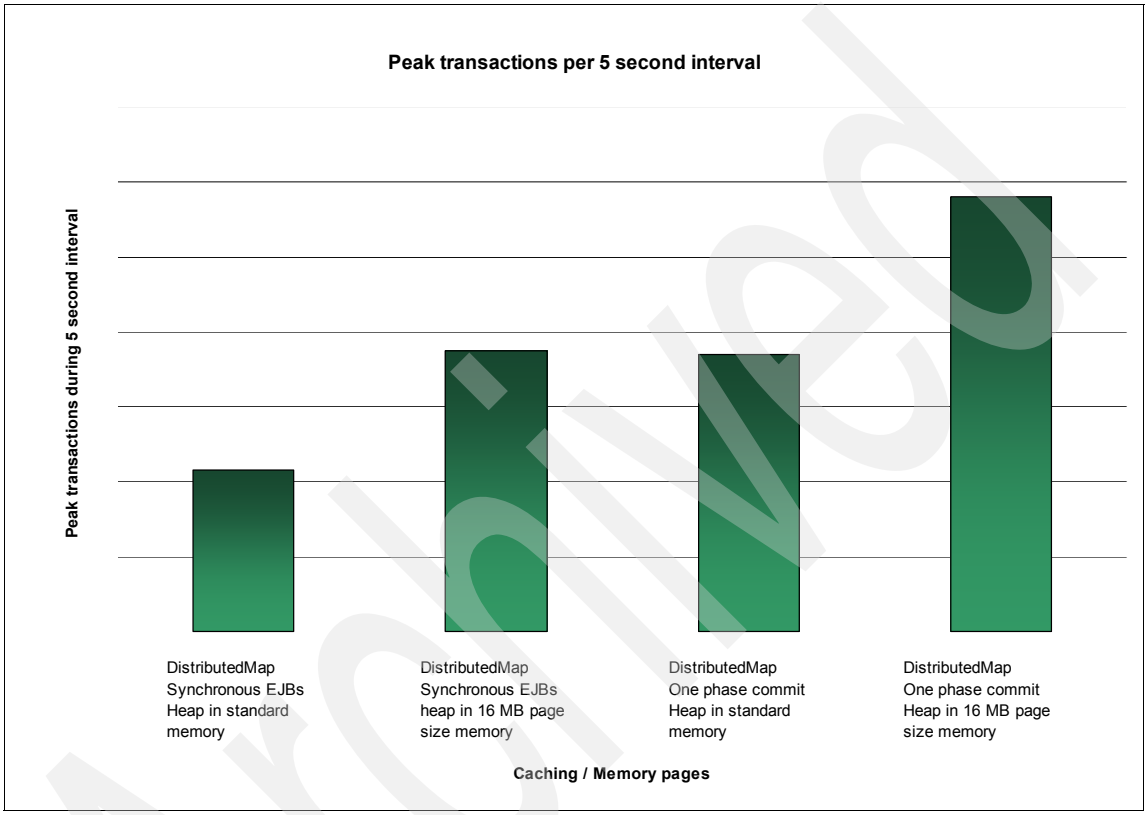


Figure 12-19 Peak transactions observed comparing runs with and without large page memory configured for the LPAR and utilized for the application server JVM heap

A cache hit is generally served in a fraction of the time of a normal request. Significant performance gains are generally achieved with Trade when leveraging the dynamic caching technology.

Additionally, we saw a significant performance benefit when the DistributedMap example design was configured under a Linux system with 1.6 GB of large (16 MB) page size memory to fulfil the JVM's maximum heap size setting.

12.7 An overview of potential tuning changes

Full performance tuning guidance is provided with the IBM WebSphere Application Server Version 6.1 product offerings. The Performance Tuning Guide is available online or can be download from:

ftp://ftp.software.ibm.com/software/webserver/appserv/library/v61/wasv610nd_tune.pdf

Here is a short list of performance suggestions and their application to System p server tuning using Red Hat Enterprise Linux AS 4 update 4 or SUSE Linux Enterprise Server 10 system images as the platforms for running WebSphere Application Server nodes.

1. Operating system
   - Linux kernel level

     2.6 kernels have many advantages over the 2.4 release.

     (Example 1) SUSE Linux Enterprise Server distributions at the SUSE Linux Enterprise Server 8 SP 2A level were relatively sensitive to context switching. To obviate this, it was possible to issue the command:

     ```
     sysctl -w sche_yiedld_scale=1
     ```

     (Example 2) Red Hat Enterprise Linux Advanced Server before 2.1 had a performance deficit with respect to memory-to-memory session replication (see 12.2.3, “HTTP Session persistence options” on page 404). Kernel levels earlier than 2.4.9-e.23 should be upgraded to avoid losing performance.

   - Resource limits

     The Linux user who has the authority to run the WebSphere Application Server needs to have adequate resource limits defined by /etc/security/limits.conf. In “Server resource limits” on page 428, we give an example of (high) ulimit levels that were beneficial in our example configuration when applied to Linux systems on all LPARs.

2. Network
   - TCP/IP timeout

     The TIME_WAIT setting determines the interval between a connection being closed and then being released. During the TIME_WAIT reopening, the connection will cost less than creating a new connection. However, if an application is characterized by the need to create new connections frequently, this interval can be reduced. For applications needing a fast release, a suggestion is to reduce the timeout to 30 seconds by issuing the command:

     ```
     echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
     ```

   - TCP/IP connection backlog

     If we anticipate high rates of incoming requests of a sufficient scale to result in disconnection failures, then it may be beneficial to adjust the backlog. A suggestion is:

     ```
     echo 3000 > /proc/sys/net/icore/netdev_max_backlog
     echo 3000 > /proc/sys/net/core/somaxconn
     ```

   - TCPIP keep alives

     The number of probes before timing out can be decreased from a default setting of nine seconds to a setting of five seconds, as follows:

     ```
     echo 5 > /proc/sys/net/ipv4/tcp_keepalive_probes
     ```

     The wait time between isAlive interval probes may be decreased from 75 seconds to 15 seconds as follows:

     ```
     echo 15 > /proc/sys/net/ipv4/tcp_keep_alive_intvl
     ```

   - Network optimized for balanced workloads

     In some scenarios, where peak loads are likely to stress network connections, the assumptions about network interface card sufficiency should be reviewed. For example, in our six application server vertical scaling scenario, we discovered that a moderate additional performance improvement was obtained by “equalizing” the connection speed back to our 300 Mbps baseline. We did this by combining three network interface cards into a bonded EtherChannel.

3. Memory
   - Allocating 16 MB pages

     The entire JVM heap size, as configured on each application server, may be adjusted to use 16 MB large page memory. In 12.6.1, “Large page sizes” on page 439, we gave the example of configuring a 1.6 GB heap size for use by application servers.

– Heap size

  The heap size for the application servers in the cluster needs to be assessed. One useful investigation is to use the **nmon** tool to create a spreadsheet that is capable of graphing memory usage trends. Additionally, we need to undertake some runs during the setup period with **verbosegc** switched on. When we did this task, we found that GCs cycled at no more than around 10% of the duration of our test runs, when we had a heap setting of around 0.5 GB. See “Heap size tuning” on page 434 for more information.

  This task needs to be evaluated for a number of test scenarios. However, production sites will tend to upgrade to maximum heap size if they encounter Out Of Memory conditions, so low heap size settings will not usually be expected on System p SMP machines. As a generalization, we should be prepared to at least double the heap size settings of application servers from the default on Linux on System p.

  Nevertheless, if large number application servers are co-resident on a single machine or LPAR, as would be the case if a large scale vertical clustering is desired, then we should remember to trim back heap sizes to fit into total available memory.

– Web container memory utilization

  We suggest increasing the thread pool settings to greater than the default and re-checking the effect in TPV. See “Web container thread pool setting” on page 430.

– EJB Container pool

  The best analysis of EJB caches and refresh intervals is made in a running test situation using TPV, as we discuss in “EJB container active pool and cache” on page 431. For the full analysis, it is worth consulting Chapter 6 of the Tuning Guide, found at:

  ftp://ftp.software.ibm.com/software/webserver/appserv/library/v61/wasv61Ond_tune.pdf

– Caching strategies

  For the entire Web application topology, it is necessary to review the caching available at the front-end environment that supplies the incoming requests and at the back end, which provides the access to database lookups or information requests. The HTTP Server provides its own cache for static content. Using DB2 9, the database server provides significant caches for SQL operations by means of prefetching and buffering. Both of these have been covered extensively in Chapter 9, “Apache Web server” on page 265 and Chapter 11, “DB2 9 data server” on page 347, respectively.

With respect to our test scenario topology in Figure 12-11 on page 422, we sized the HTTP Server to be sufficient to route tens of thousands of requests without loading CPU or memory utilization beyond around 10%. By a similar strategy, the database was made to respond in less than a half-second fashion to queries while sustaining up to 25% utilizations. So in this situation, the front- and back-end servers were contrived to be of large throughput capacity (which means that they were “oversized”) compared to the application servers. This helps ensure that all bottlenecks would occur inside the application server or application server cluster. Caching, and other resource characteristics, inside the application servers, is the subject of the present chapter.

– Programmatic caching

Internal caching can be exploited in Java enterprise programs using the Distributed Map API with suitable elements configured into the WebSphere internal cache as part of the Web container description using the cachespec.xml file. This needs to be defined for the application and deployed (onto each node). We have given an example of caching services in 12.6.2, “Dynamic caching services” on page 442 that used this technique. When accompanied with a large page memory setup, we saw a useful increase in throughput.

4. Disk

– Tranlogs

Each application server implements a local directory for storing the current transactions. For example, our w590lp1-1 cluster member uses the directory /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/tranlog to hold these volatile files, assuming a default installation of WebSphere Application Server ND 6.1. [4]

Because they are very frequently updated whenever the application server is placed under load, there may be performance improvements to be gained by mounting the tranlog directory as a remote file system with fast performance characteristics. So we might implement the tranlog mount point physically on a SCSI tower or fast storage partition.

– JVM logs

Relative to the profile directory location of the instance, there are important logs held for the application server runtime process JVM. The runtime logging is divided between system output and system error, in the form files called SystemOut.log and SystemErr.log.

[4] This assumes that the cluster members have been provided by installing WebSphere ND on the 590 LPARs. Potentially, the installation may have used ND only for the deployment manager installation on a separate machine and used WebSphere Application Server base product for the installation on each LPAR of the 590. If this had been the installation route, then the default directory for tranlogs would be /opt/IBM/WebSphere/AppServer/profiles/default/tranlog.

We would not usually expect any lines to be written into SystemErr.log because this indicates a condition that should be analyzed before continuing the run. By contrast, some continual growth of SystemOut.log is perfectly useful, normal behavior. Since the log is regularly appended to, there may be some benefit in preparing a special file system for it.

– XML configuration files

  There are a number of XML files to be read in at cluster start time and at application launch time, in the directory called *config*. A large enterprise application may use about one thousand XML files. In our installation, there were 565 files. Looking at them, it becomes apparent that the EAR files can be several hundred of kB, but the majority of the files occupy less than 100 KB.

  The configuration files should be largely unchanged during the life of a running application. They will change when a configurational change is made as a result of administering the cluster or making a local change to the application server. In our installation, the total filestore used by this largely static content is 121 MB. However, the more application server profiles are configured and running, the greater will be the number of these configuration files. At some stage in scaling up an application server farm, there is the possibility that accessing the configuration files will contribute to a file system performance degradation.

  A series of test runs can be set up to see how much of a benefit can be gained from special storage arrangements for the three subdirectories: tranlogs, logs (subdivided into cluster members), and config. These subdirectories are to be found relative to the instance profile's root, for example, /opt/IBM/WebSphere/AppServer/profiles/AppSrv01 in the case of the default installation of ND.[5]

– Adding hard disks

  A common practice on System p servers is to ensure that there is an additional internal SCSI disk available, in addition to the disk used to boot and provide the operating system. Where this is available, for example, a /dev/sdb or /dev/sdc entry is available for automounting according to /etc/fstab, it is a best practice to set up a specific mount point, /opt/IBM/WebSphere, to hold all the disk contents used by any stand-alone or clustered application servers. Holding these contents separately from the operating system is recommended. In practice, one extra internal SCSI disk for the application server(s) running on the LPAR is often found to be an adequate solution to file contention issues, providing each node is on a machine or LPAR that has one extra disk at its disposal.

[5] In the case of a default installation based on the IBM WebSphere Application Server base product, this default location is /opt/IBM/WebSphere/AppServer/profiles/default.

This solution may be insufficient for architectures that are more locally disk intensive than usual. A particular case would be applications that require frequent passivation of stateful session beans to local disk space.

Additionally, the guidance presented in this section cannot be applied to servers or LPARs where the back-end database is co-resident. We have assumed a separate remotely accessible database using a JDBC driver capable of network access throughout this review. If there is a co-located database server, then the advice for database file system tuning takes precedence over any storage suggestion in this chapter.

12.8 Summary

The changes that need to be made to the system image to ensure high levels of scalability and performance are:

- At each LPAR, set the ulimit configuration to realistic levels for the processing required. In particular, the number of file handles usually needs to be at least several thousand to cope with peak transactional loads.
- Obtain performance traces and CPU / memory logs, such as those from **nmon**, for representative runs of the application. Together with information from the Performance Advisor, these will help indicate whether resource tuning adjustments are possible.
- Ensure that the Web container's thread pool settings are sufficient to allow the best possible throughput in serving the Web address requests emanating from the HTTP Server, so as to minimize queuing of requests. Consider carefully whether to unrestrict the thread pool so that it is allowed to overflow. If possible, stage sufficient tests to be able to define higher limits to the minimum and maximum pool size, providing that this does not present too great a stress to the application server.
- Set the application's heap to provide the best balance between requiring frequent garbage collections, because heap is exhausted too quickly and lengthy garbage collection times result. On System p SMP LPARs, setting the maximum heap size to 512 MB or greater is usually efficient using the 64-bit WebSphere implementation.
- Consider using the large page setting for use of memory in 16 MB pages. Remember to calculate a large enough value for **shmmax** if you are setting up large page memory for a Java heap.
- When starting at the application stage, make extensive use of the com.ibm.websphere.cache.DistributedMap exploitation of dynamic caching support to servlets and objects running in WebSphere Application Server

V6.1. Large page size memory to make up the Java heap provides increased performance benefits if object caching is adopted.

- Check and, if necessary, amend the settings of connection pools as indicated by test runs under load while running Tivoli Performance Viewer and Performance Advisor.

Changes to the configuration cannot lead to greater performance efficiency if the critical resources are exhausted by the envisaged load. The critical resources are:

- Total processor count
- Memory
- Network bandwidth

We found that a total of six processors were sufficient to respond to a peak transactional load using the Trade 6 EJB testing scenario at up to around 3000 transactions per second (721 transactions every five seconds) before the application server was driven into stress. As measured, the CPU was driven to near saturation levels. Memory was sufficient at about 300 MB and the maximum Java heap could be tuned down to less than 400 MB if necessary.

In our particular test scenario, we discovered that there was no advantage, and possibly a slight deterioration, by adding application servers so as to scale vertically. However, dividing up the same resources across three LPARs running single application servers per LPAR, there was a slight increase in total transactional throughput to around 3800 transactions per second. Adding an extra application server to each LPAR reduced peak transactional performance, however.

Network effects were present, although not overstretching the capacity of a 100 Mbps Ethernet card per LPAR. In the case of a single LPAR, to compare against three separate LPARs each with individual Ethernet cards, we found that bonding the three original Ethernet cards into a single Etherchannel configuration at nominally 300 Mbps gave an improvement over a single card.

Our recommendation would be not to consider using vertical scaling where there are less than four cores to be distributed across, unless there is some clearly identifiable resource that is restricted by running in a single process. Otherwise, there may be performance gains to be obtained by running in a WebSphere cluster, along with improved availability and the ability to fail over in the event of a fault.

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 461. Note that some of the documents referenced here may be available in softcopy only.

- *Advanced POWER Virtualization on IBM System p5: Introduction and Configuration*, SG24-7940
- *Advanced POWER Virtualization on IBM eServer p5 Servers: Architecture and Performance Considerations*, SG24-5768
- *Advanced POWER Virtualization on IBM eServer p5 Servers: Architecture and Performance Consideration*, SG24-5768
- *Deployment Guide Series: TotalStorage Productivity Center for Data*, SG24-7140
- *Effective System Management Using the IBM Hardware Management Console for pSeries*, SG24-7038
- *Experience J2EE! Using WebSphere Application Server V6.1*, SG24-7297
- *IBM BladeCenter JS21 Technical Overview and Introduction*, REDP-4130
- *IBM eServer BladeCenter and Topspin InfiniBand Switch Technology, REDP-3949*
- *IBM IntelliStation POWER 185 Technical Overview and Introduction*, REDP-4135
- *IBM IntelliStation POWER 285 Technical Overview and Introduction*, REDP-4078
- *IBM System p5 505 and 505Q Technical Overview and Introduction*, REDP-4079
- *IBM System p5 510 and 510Q Technical Overview and Introduction*, REDP-4136
- *IBM System p5 520 and 520Q Technical Overview and Introduction*, REDP-4137

- *IBM System p5 550 and 550Q Technical Overview and Introduction*, REDP-4138
- *IBM System p5 560Q Technical Overview and Introduction*, REDP-4139
- *IBM System p5 570 Technical Overview and Introduction*, REDP-9117
- *IBM System p5 590 and 595 Technical Overview and Introduction*, REDP-4024
- *IBM System p5 185 Technical Overview and Introduction*, REDP-4141
- *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039
- *Integrated Virtualization Manager on IBM System p5*, REDP-4061
- *Introduction to InfiniBand for IBM eServer pSeries, REDP-4095*
- *Linux Handbook A Guide to IBM Linux Solutions and Resources*, SG24-7000
- *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039
- *Patterns: Service-Oriented Architecture and Web Services*, SG24-6303
- *Web Services Handbook for WebSphere Application Server 6.1*, SG24-7257
- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392

Other publications

These publications are also relevant as further information sources:

- Couchman, *Oracle 8 Certified Professional. DBA Certification Exam Guide*, Osborne Publishing, 1999, 0072120878

Online resources

These Web sites are also relevant as further information sources:

- Alternative PHP Cache

  `http://pecl.php.net/package/APC`

- Apache

  `http://httpd.apache.org/docs/2.0/mod/core.html#loglevel`

- Apachetop

  `http://www.webta.org/projects/apachetop/`

- “Automatic Performance Diagnostics” in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/diagnsis.htm#sthref419

- “Automatic Workload Repository” in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/autostat.htm#sthref362

- Awstat

  http://awstats.sourceforge.net/

- BladeCenter JS21

  http://www.redbooks.ibm.com/abstracts/redp4130.html?Open

- *Choosing an I/O Scheduler for Red Hat Enterprise Linux 4 and the 2.6 Kernel,* found at:

  http://www.redhat.com/magazine/008jun05/features/schedulers/

- Common Public License

  http://www.ibm.com/developerworks/opensource/library/os-cplfaq.html

- “Configuring Dispatchers” in *Oracle Database Net Services Administrator's Guide 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/network.102/b14212/dispatcher.htm#NETAG012

- *Configuring Oracle Server for VLDB*, found at:

  http://www.miracleas.com/BAARF/0.Millsap1996.08.21-VLDB.pdf

- *Diagnosing and Resolving Performance Problems Using ADDM,* found at:

  http://www.oracle.com/technology/obe/10gr2_db_single/manage/addm/addm_otn.htm

- “Dynamic Performance Views” in *Oracle Database Reference 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14237/dynviews_part.htm#i403961

- eAccelerator

  http://www.eaccelerator.net/

- Eclipse

  http://www.eclipse.org

- GNU

  http://www.gnu.org/philosophy/free-sw.html

- Hardware Management Console

  http://www.redbooks.ibm.com/abstracts/sg247038.html?Open

- “Instance Tuning Using Performance Views” in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/instance_tune.htm#i35312

- Integrated Virtualization Manager

  http://www.ibm.com/developerworks/linux/library/l-pow-ivm/

- IntelliStation 185

  http://www.redbooks.ibm.com/abstracts/redp4135.html?Open

- IntelliStation 285

  http://www.redbooks.ibm.com/abstracts/redp4078.html?Open

- JDBC-ODBC bridge enhancements

  http://java.sun.com/j2se/1.5.0/docs/guide/jdbc/bridge.html

- Jumbo Frames resource materials

  http://www.small-tree.com/resources/pdfs/jumbo_white_paper.pdf

- Lighttpd

  http://www.lighttpd.net/
  http://trac.lighttpd.net/trac/wiki/PoweredByLighttpd

- Linux 2.6

  http://www.ibm.com/developerworks/linux/library/l-inside.html

- Linux Distributions

  http://distrowatch.com/
  http://www.ibm.com/systems/p/linux/

- Linux on POWER

  http://www-1.ibm.com/partnerworld/pwhome.nsf/weblook/pat_linux_learn_tech.html

- Linux POWER Applications

  http://www-03.ibm.com/systems/linux/power/apps/all.html

- Linux Standard Base

  http://www.linuxbase.org

- Linux Technology Center

  http://www.ibm.com/developerworks/linux/ltc/

- “Managing Oracle Database Processes” in *Oracle Database Administrator's Guide 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14231/manproc.htm#i1010000

- “Memory Architecture” in *Oracle Database Concepts 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14220/memory.htm#i12483

- “Memory Configuration and Use” in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/memory.htm#i49320

- Modruby

  http://modruby.net/

- Open Firmware

  http://www.openfirmware.org

- “Optimizing SQL Statements” in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/part4.htm#sthref1059

- *Oracle Database Application Developer's Guide - Fundamentals 10g Release 2 (10.2)*, found at:

  http://download-east.oracle.com/docs/cd/B19306_01/appdev.102/b14251/toc.htm

- *Oracle Database 10g: The Top 20 Features for DBAs. Week 6: Automatic Workload Repository,* found at:

  http://www.oracle.com/technology/pub/articles/10gdba/week6_10gdba.html

- *Oracle Database 10g: The Top 20 Features for DBAs. Week 18: ADDM and SQL Tuning Advisor*, found at:

  http://www.oracle.com/technology/pub/articles/10gdba/week18_10gdba.html

- p5-185

  http://www.redbooks.ibm.com/abstracts/redp4141.html?Open

- p5-505 and p5-505Q

  http://www.redbooks.ibm.com/abstracts/redp4079.html?Open
- p5-510 and p5-510Q

  http://www.redbooks.ibm.com/abstracts/redp4136.html?Open
- p5-520 and p5-520Q

  http://www.redbooks.ibm.com/abstracts/redp4137.html?Open
- p5-550 and p5-550Q

  http://www.redbooks.ibm.com/abstracts/redp4138.html?Open
- p5-560Q

  http://www.redbooks.ibm.com/abstracts/redp4139.html?Open
- p5-570

  http://www.redbooks.ibm.com/redpieces/abstracts/redp9117.html?Open
- p5-590 and p5-595

  http://www.redbooks.ibm.com/redpieces/abstracts/redp4024.html?Open
- “Performance Considerations for Shared Servers” in *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*, found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14211/build_db.htm#i22941
- “Process Architecture” in *Oracle Database Concepts 10g Release 2 (10.2),* found at:

  http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14220/process.htm#i21263
- *Providing OLAP to User-Analysts: An IT Mandate*, found at:

  http://dev.hyperion.com/resource_library/white_papers/providing_olap_to_user_analysts.pdf
- *Reliability, Availability, Serviceability (RAS) for Linux on POWER*, found at:

  http://www6.software.ibm.com/cgi-bin/pwdown/public/httpdl/linux/LoP_RAS.pdf
- SNMP Apache Module

  http://mod-apache-snmp.sourceforge.net/english/index.htm
- Swingbench

  http://www.dominicgiles.com/swingbench.php

- *Towards Linux 2.6: A look into the workings of the next new kernel*, found at:

  http://www.ibm.com/developerworks/linux/library/l-inside.html
- *Tuning Oracle Database 10g for ext3 file systems,* found at:

  https://www.redhat.com/magazine/013nov05/features/oracle/
- Turck MMCache

  http://turck-mmcache.sourceforge.net/
- "Using Statspack" in *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2),* found at:

  http://download-east.oracle.com/docs/cd/B10501_01/server.920/a96533/statspac.htm
- Virtualization

  http://www-03.ibm.com/systems/p/apv/
- XML Guide and the XQuery Reference

  http://ibm.com/software/data/db2/udb/support/manualsv9.html
- Virtual I/O Server

  http://techsupport.services.ibm.com/server/vios/documentation/perf.html
- Webalizer

  http://www.mrunix.net/webalizer/
- Zend Optimizer

  http://www.zend.com/products/zend_optimizer

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Tuning Linux OS on IBM System p: The POWER of Innovation

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

IBM®

Tuning Linux OS on System p
The POWER of Innovation

Redbooks

Linux kernel tuning

DB2, WebSphere, and Oracle tuning

SystemTap examples

This IBM Redbooks publication describes what you can do to improve and maximize the performance of your business server applications running on System p hardware with the Linux operating system.

It describes how to improve the performance of the System p hardware, the operating system, and specific server applications.

The book is divided into three parts:

Part 1 explains the technology implemented in the major subsystems in System p servers and shows what settings can be selected or adjusted to obtain the best performance.

Part 2 describes the performance aspects of the operating systems: Red Hat Enterprise Linux, SUSE LINUX Enterprise Server, and IBM Virtual I/O Server.

Part 3 introduces the performance monitoring tools that are available to users of System p servers. Part 4 shows you how to analyze your system to find performance bottlenecks and what to do to eliminate them. Part 5 examines specific performance characteristics of specific server applications.

This book is targeted at people who configure POWER processor based servers running Linux and seek to maximize performance. Some knowledge of servers is required. Skills in performance tuning are not assumed.

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

SG24-7338-00 ISBN 0738486434