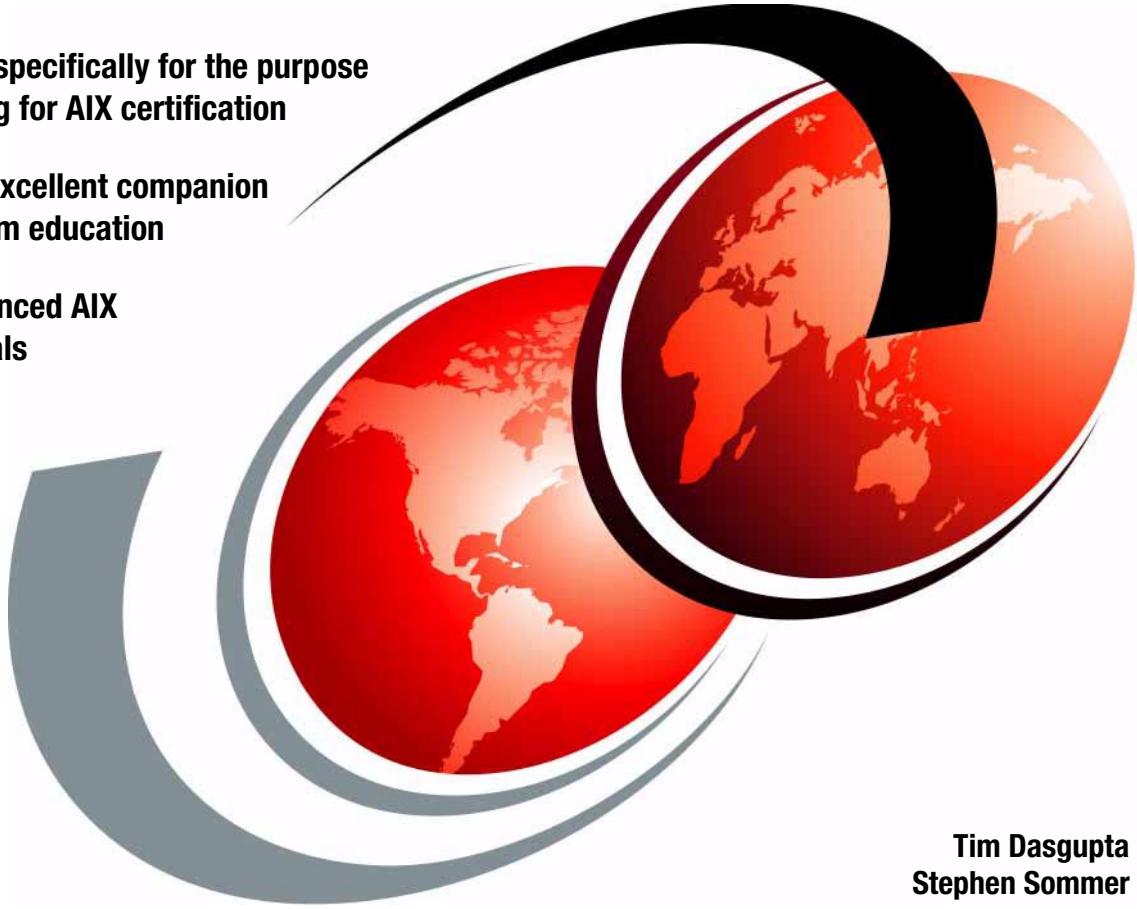
IBM

IBM Certified Specialist

IBM eserver Certification Study Guide - AIX 5L Performance and System Tuning

- Developed specifically for the purpose of preparing for AIX certification
- Makes an excellent companion to classroom education
- For experienced AIX professionals



Tim Dasgupta
Stephen Sommer

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM @server Certification Study Guide - AIX 5L Performance and System Tuning

December 2002

SG24-6184-01

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

Second Edition (December 2002)

This edition applies to AIX 5L Version 5.1 (5765-E61) and subsequent releases running on an IBM @server pSeries or RS/6000 server.

Contents

Figures

Tables

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

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

Trademarks

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

AFP™	Micro Channel®	Redbooks™
AFS®	Perform™	Redbooks(logo)™ 
AIX®	PowerPC®	RS/6000®
AIX 5L™	PowerPC 601®	Sequent®
Balance®	PowerPC 604™	SP™
DFS™	PowerPC Reference Platform®	TotalStorage™
@server	pSeries™	Versatile Storage Server™
IBM®	PTX®	
LoadLeveler®	QMF™	

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

Domino™	Lotus®	Word Pro®

The following terms are trademarks of other companies:

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

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

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

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

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

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

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

Preface

The AIX and IBM @server pSeries Certifications offered through the Professional Certification Program from IBM are designed to validate the skills required of technical professionals who work in the powerful and often complex environments of AIX and IBM @server pSeries. A complete set of professional certifications is available. It includes:

- IBM Certified AIX User
- IBM Certified Specialist - Business Intelligence for RS/6000
- IBM Certified Specialist - Domino for RS/6000
- IBM @server Certified Specialist - p690 Solutions Sales
- IBM @server Certified Specialist - p690 Technical Support
- IBM @server Certified Specialist - pSeries Sales
- IBM @server Certified Specialist - pSeries AIX System Administration
- IBM @server Certified Specialist - pSeries AIX System Support
- IBM @server Certified Specialist - pSeries Solution Sales
- IBM Certified Specialist - RS/6000 SP and PSSP V3
- IBM Certified Specialist - Web Server for RS/6000
- IBM @server Certified Specialist - pSeries HACMP for AIX
- IBM @server Certified Advanced Technical Expert - pSeries and AIX 5L

Each certification is developed by following a thorough and rigorous process to ensure the exam is applicable to the job role and is a meaningful and appropriate assessment of skill. Subject matter experts who successfully perform the job participate throughout the entire development process. They bring a wealth of experience into the development process, making the exams much more meaningful than the typical test that only captures classroom knowledge and ensuring the exams are relevant to the *real world*. Thanks to their effort, the test content is both useful and valid. The result of this certification is the value of appropriate measurements of the skills required to perform the job role.

This IBM Redbook is designed as a study guide for professionals wishing to prepare for the AIX 5L Performance and System Tuning certification exam as a selected course of study in order to achieve the IBM @server Certified Advanced Technical Expert - pSeries and AIX 5L certification.

This IBM Redbook is designed to provide a combination of theory and practical experience needed for a general understanding of the subject matter. It also provides sample questions that will help in the evaluation of personal progress

and provide familiarity with the types of questions that will be encountered on the exam.

This publication does not replace practical experience, nor is it designed to be a stand-alone guide for any subject. Instead, it is an effective tool that, when combined with education activities and experience, can be a very useful preparation guide for the exam.

For additional information about certification and instructions on how to register for an exam, visit our Web site at:

http://www.ibm.com/certify

The team that wrote this redbook

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

Tim Dasgupta is an IBM Certified AIX Advanced Technical Expert (CATE). He works as a Senior Systems Architect at IBM Global Services in Canada. He has over eight years of experience in the areas of AIX, RS/6000, and pSeries. He is currently the Team Leader of Midrange Architecture Group in Montreal, Canada.

Stephen Sommer is an IBM Certified AIX Advanced Technical Expert (CATE), AIX Version 4.3.3 and 5.1. He works as a Senior IT Specialist at Faritec Services, an IBM Business Partner in Johannesburg, South Africa. He has eight years of experience in Midrange Support for AIX, RS/6000, and pSeries, both in South Africa and the United Kingdom.

The authors of the first edition are:

Thomas Herlin IBM Denmark

André de Klerk IBM South Africa

Thomas C. Cederlöf IBM Sweden

Tomasz Ostaszewski Prokom Software SA in Poland

The project that produced this publication was managed by:

Scott Vetter IBM Austin

Special thanks to:

Shannan L DeBrule IBM Atlanta

Darin Hartman Program Manager, AIX Certification

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

Jesse Alcantar, Greg Althaus, Larry Brenner, Shawn Mullen, Brian Nicholls, Greg Flaig
IBM Austin

Michelle Page-Rivera
IBM Atlanta

John Hance and Tony Steel
IBM Australia

Edward Geraghty
IBM Boston

Adnan Ikram
IBM Pakistan

Christopher Snell
IBM Raleigh

Stephen Atkins and Peter Mayes
IBM U.K.

Karl Borman
ILS Austin

Malin Cederberg and Robert Olsson
ILS Sweden

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493

1

Certification overview

This chapter provides an overview of the skill requirements needed to obtain an IBM Advanced Technical Expert certification. The following chapters are designed to provide a comprehensive review of specific topics that are essential for obtaining the certification IBM @server Certified Advanced Technical Expert - pSeries and AIX 5L.

This level certifies an advanced level of AIX knowledge and understanding, both in breadth and depth. It verifies the ability to perform in-depth analysis, apply complex AIX concepts, and provide resolution to critical problems, all in a variety of areas within AIX.

1.1 Certification requirements

To attain the IBM @server Certified Advanced Technical Expert - pSeries AIX 5L certification, you must pass four tests.

One test is the prerequisite in either pSeries AIX System Administration or pSeries AIX System Support. The other three tests are selected from a variety of pSeries and AIX topics. These requirements are explained in greater detail in the sections that follow.

1.1.1 Required prerequisite

Prior to attaining the IBM @server Certified Advanced Technical Expert - pSeries and AIX 5L certification, you must be certified as either an:

- IBM @server Certified Specialist - pSeries AIX System Administration

  or

- IBM @server Certified Specialist - pSeries AIX System Support

1.1.2 Recommended prerequisite

A minimum of six to 12 months experience in performing in-depth analysis and applying complex AIX concepts in a variety of areas within AIX is a recommended prerequisite.

1.1.3 Information and registration for the certification exam

For the latest certification information, see the following Web site:

http://www.ibm.com/certify

1.1.4 Core requirements

Select three of the following exams. You will receive a Certificate of Proficiency for tests when passed.

AIX 5L Installation and System Recovery

Test 233 was developed for this certification.

Preparation for this exam is the topic of *IBM @server Certification Study Guide - AIX Installation and System Recovery*, SG24-6183.

AIX 5L Performance and System Tuning

The following objectives were used as a basis when the certification test 234 was developed. Some of these topics have been regrouped to provide better organization when discussed in this publication.

Preparation for this exam is the topic of this publication.

Section I - Measurements and baseline

Section I cover measurements and baseline.

1. Determine successful operating levels for stable/comparable system.
   a. Run the **oslevel** command to determine current OS version.
   b. Run the **lslpp** command to list installed software.
   c. Run **errpt** to determine if there are any unresolved hardware problems.
2. Take snapshot/baseline of system.
   a. Install perfagent.tools fileset.
   b. Install bos.perf fileset.
   c. Install bos.adt.samples fileset.
   d. Verify that you have the latest filesets for all of the above steps; if not, obtain the latest and install them.
   e. Identify time to take performance measurements.
   f. Run **perfpmr** to gather operating statistics.
   g. Run **no**, **nfso**, **vmtune**, or **schedtune** to gather tuning parameters if appropriate.
   h. Run **wlm** (Workload Manager) to gather tuning parameters.
   i. Run **topas** (Performance Monitoring Tool) to gather tuning parameters.
3. Establish performance policies for the new system (system to be tuned).
   a. Determine which measurements are relevant.
   b. Define (using Service Level Agreements) what a well-behaved system is.

Section II - Performance monitoring and analysis

Section II covers performance monitoring and analysis.

1. Determine if CPU bound.
   a. Run **vmstat**, **iostat**, **sar**, or **topas** to gather CPU utilization information.
   b. Add %user and %sys to determine if CPU bound.
2. If CPU bound, run additional tools like **tprof** and **ps** to determine which process is causing the bottleneck. Determine if memory bound.

a. Run **vmstat** to gather memory utilization information.

b. Look at pi and po output columns to determine if memory bound.

c. Use /proc to determine if the system is memory bound.

d. If memory bound, run tools like **svmon** and **ps** to determine which process is causing the bottleneck.

3. Determine if I/O bound.

   a. Run **iostat** or **topas** to gather disk I/O utilization information.

   b. Look at %iowait and %tm_act to determine if I/O bound.

   c. If I/O bound, run tools like **filemon**, **fileplace**, and **lslv** to determine which process, disk, or adapter is causing the bottleneck.

4. Determine if network bound.

   a. Run **netpmon**, **netstat**, or **nfsstat** to gather network utilization information.

   b. Run **entstat** or **fddistat** to gather adapter-specific information.

   c. Look at output to determine if network bound.

   d. If network bound, run tools like **no**, **netpmon**, **nfso**, or **lsattr** to assist in determining the cause of the bottleneck.

Section III - Tuning

Section III covers tuning.

1. Tune application.

   Analyze application with profiling tools like **tprof**, **gprof**, and **prof** to identify potential code optimization opportunities.

2. Tune CPU.

   a. Balance system workload.

   b. Tune scheduler using nice/renice to give different priorities to running processes (prevent CPU hogging).

   c. Tune scheduler algorithm using schedtune to fine tune priority formulas, setting timeslices.

3. Tune memory.

   a. Add more real memory to increase available memory.

   b. Tune virtual memory manager using **vmtune** command.

   c. Tune memory overcommitment algorithm using **schedtune** command to prevent thrashing.

4. Tune I/O.

a. Place data on physical disks appropriately (striping) using tools such as **reorgvg**, **migratepv**, or **chlv**.

b. Use **vmtune** to modify file system, lvm, and paging parameters.

c. Use **chgsys** to tune high/low water marks for pending write I/Os (I/O pacing).

d. Place JFS logs on different physical disks from file system data.

e. Defragment files; make them contiguous.

5. Tune network.

a. Tune transmit and receive queues using **chdev**.

b. Use **no** to tune TCP/IP options.

c. Use **ifconfig** or **no** to adjust MTU size appropriate for your network topology.

d. Use /etc/netsvc.conf or NSORDER environment variable to override default order for name resolution.

e. Tune NFS options (if appropriate).

Section IV - System sizing

Section IV covers system sizing.

1. Gather additional information.

a. Run performance monitoring tools to gather additional information.

b. Analyze data to predict future system requirements.

2. Use sizing tools.

Analyze results of sizing tool.

AIX 5L Problem Determination Tools and Techniques

Test 235 was developed for this certification.

Preparation for this exam is the topic of *IBM @server Certification Study Guide - AIX Problem Determination Tools and Techniques*, SG24-6185.

AIX 5L Communications

Test 236 was developed for this certification.

Preparation for this exam is the topic of *IBM @server Certification Study Guide - AIX Communications*, SG24-6186.

pSeries HACMP for AIX

Test 187 was developed for this certification.

Preparation for this exam is the topic of *IBM @server Certification Study Guide - pSeries HACMP for AIX*, SG24-6187.

RS/6000 SP and PSSP V3.1

Test 188 was developed for this certification.

Preparation for this exam is the topic of *IBM @server Certification Study Guide - RS/6000 SP*, SG24-5348.

p690 Technical Support

Test 195 was developed for this certification.

An IBM Redbook is planned for first quarter 2003 on this subject.

1.2 Certification education courses

Courses are offered to help you prepare for the certification tests. For a current list, visit the following Web site, locate your test number, and select the education resources available:

http://www.ibm.com/certify/tests/info.shtml

2

Performance tuning: Getting started

In this chapter, the following topics are covered:

- Introduction to concepts and tools
- Performance tuning flowchart

In general, the performance tuning issues can be divided into two areas:

- System management
- Application development

The application developer will usually view performance more from a user's perspective than a system perspective; for example, the user response time and system interactions are the concerns addressed during the design phase, not the overall system performance. This aspect of performance tuning, optimization of code, is outside the scope of this publication. This publication focuses on the system management aspects.

Many of the commands introduced in this chapter are discussed in detail throughout this publication (as referenced).

2.1 Introduction to concepts

Performance management, from a system management point of view, is usually concentrated on the allocation of existing resources, but also includes allocation of additional resources and establishment of system policies. Therefore, performance tuning can be defined as the application and allocation of resources to best meet the defined requirements and goals.

From this definition of performance tuning, the following list provides a set of tasks that are part of performance tuning:

1. Identify the workload.

   If the system to tune is a workstation, then the most probable goal is fast response time.

   If the system is a multiuser environment, the goal is to maximize throughput within a given response time or minimize response time under a consistent workload.

   If the system is a server, maximizing throughput for a given response time is usually the goal.

2. Define and prioritizing goals.

   Before starting a tuning exercise, you need to have clear goals in mind. In other words, how will you know when you have finished tuning the system? Bear in mind that response time and throughput are not the same thing and that you need to focus on one or the other for each application. It is also important to realize that tuning is a process of compromise—that you will likely be taking resources away from one application to give to another. A clear understanding of the relative priorities that operate in your environment is also an essential prerequisite to tuning your system.

3. Identify the required resources.

   Performance of a given workload is determined by the availability and speed of certain critical resources. Resources can be divided into two areas: Physical resources and logical resources. Table 2-1 provides examples of hardware resources with their logical resources.

   From the points discussed, steps 1 through 3 are part of planning and researching.

Table 2-1 Hardware resources and logical resources

Hardware resource	Logical resource
CPU	Process time slice

Hardware resource	Logical resource
Memory	Page frame
	Stacks
	Buffers
	Queues
	Tables
Disk space	Logical volumes
	File systems
	Files
Communication lines	Packets
	Channels

4. Minimize resource requirements.

   Resource requirements can be minimized by using optimized code, organizing data efficiently, rescheduling low-prioritized jobs, making the right choice when to use remote resources, and so on. This is the stage where the actual hands-on tuning will occur.

5. Control allocation of resources.

   Resources to control include disk space and process priority control. Disk space for users, or groups of users, can easily be managed with a quota, and process priority can be handled with the Workload Manager or by manipulating the scheduler.

6. Repeat steps 3 to 5 as necessary.
7. Add additional resources as required.

In the following section, a common performance tuning flowchart will be briefly discussed.

2.2 CPU performance overview

When investigating a performance problem, CPU constraint is probably the easiest to find. That is why most performance analysts start by checking for CPU constraints, and then work their way through the flowchart shown in Figure 2-1 on page 10.

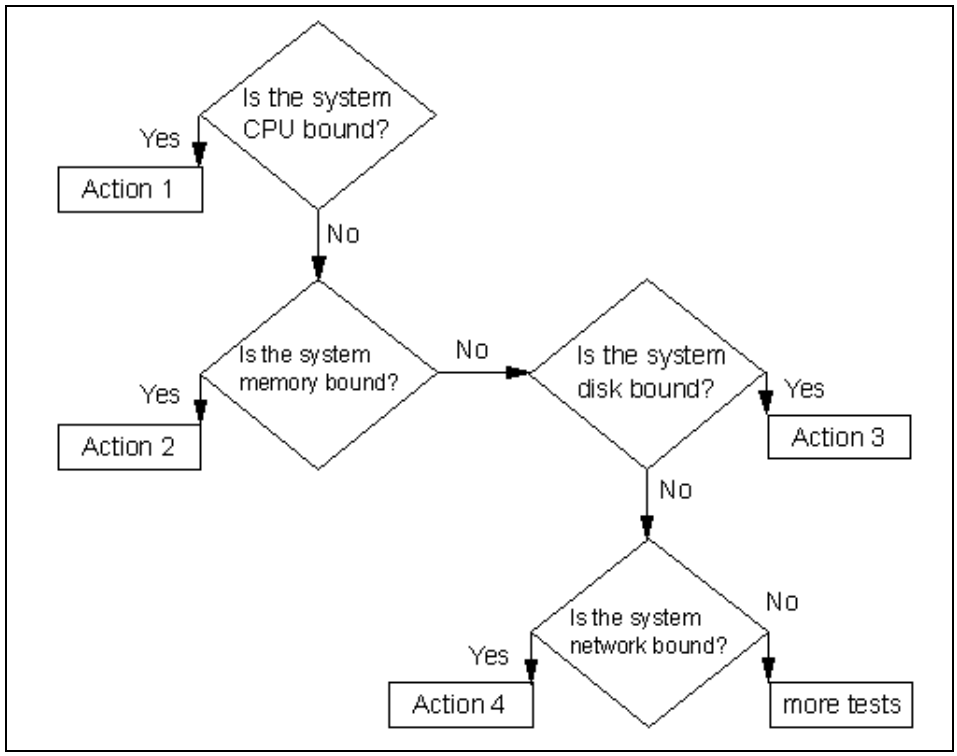


Figure 2-1 General performance tuning flowchart

If a system is CPU bound, investigation should focus on the two entities using the CPU: Processes and threads. The CPU's basic work unit is the thread, so a process must have at least one thread. Commonly (on AIX Version 4), a process is multi-threaded, which means that a process can use multiple threads to accomplish its task. In Figure 2-2 on page 11, the relationship between processes and threads is symbolized.

When initiating a process, the first resource to be allocated is a slot in the process table; before this slot is assigned, the process is in SNONE state. While the process is undergoing creation (waiting for resources [memory] to be allocated) it is in SIDL state. These two states are together called the I state.

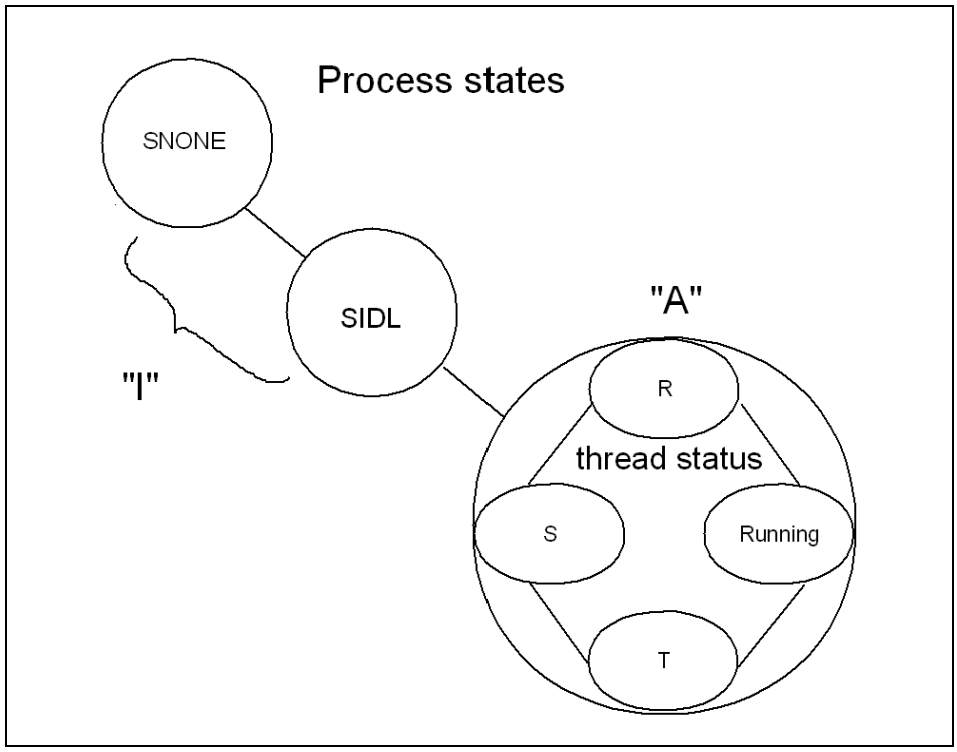


Figure 2-2 Process state

When a process is in the A state, one or more of its threads are in the R state. This means they are ready to run. A thread in this state has to compete for the CPU with all other threads in the R state. Only one process can use the CPU at any given time.

If a thread is waiting for an event or for the I/O, the thread is said to be sleeping, or in the S state. When the I/O is complete, the thread is awakened and placed in the ready-to-run queue.

If a thread is stopped with the SIGSTOP signal (to be awakened with the SIGCONT signal), it is in the T state while suspended.

Manipulating the run queue, the process and thread dispatcher, and priority calculation are all ways to tune (and misstune, if not carefully done) the CPU. The run queue and how to decide which thread is to be prioritized is discussed in Chapter 6, “Performance management tools” on page 205.

When tuning the CPU, you need to know what can be tuned on a process level and what can be tuned on a thread level and choose accordingly. Table 2-2 on

page 12 provides a list that associates several process-related properties with thread-related properties.

Table 2-2 Processes and threads

Process properties	Thread properties
PID and PGID	TID
UID and GID	Stack
Environment	Scheduling policy
Cwd	Pending signals
File descriptors	Blocked signals

When working in the area of CPU performance tuning, you should use historical performance information for comparison reasons. Usually, performance has subjective view points. To avoid confusion, hard copies of performance statistics, from a time when users did not report poor system performance, should be filed. A very useful tool for this task is the **sar** command.

2.2.1 The sar command

Two shell scripts, /usr/lib/sa/sa1 and /usr/lib/sa/sa2, are structured to be run by the **cron** command and provide daily statistics and reports. Sample stanzas are included (but commented out) in the /var/spool/cron/crontabs/adm crontab file to specify when the cron daemon should run the shell scripts. The sa1 script creates one output file each day and the sa2 script collects data and saves the data for one week. Another useful feature of **sar** is that the output can be specific about the usage for each processor in a multiprocessor environment, as seen in the following output. The last line is an average output.

```
# sar -P ALL 2 1

AIX client1 3 4 000BC6DD4C00    07/06/00

14:46:52 cpu    %usr    %sys    %wio   %idle
14:46:54  0        0       0       0     100
          1        0       1       0      99
          2        0       0       0     100
          3        0       0       0     100
          -        0       0       0     100
```

More information on the **sar** command can be found in 3.1, “The sar command” on page 40.

Occasionally, the time spent in an application execution or an application startup can be useful to have as reference material. The **time** command can be used for this.

2.3 The time command

Use the **time** command to understand the performance characteristics of a single program and its synchronous children. It reports the real time, that is, the elapsed time from beginning to end of the program. It also reports the amount of CPU time used by the program. The CPU time is divided into user and sys components. The user value is the time used by the program itself and any library subroutines it calls. The sys value is the time used by system calls invoked by the program (directly or indirectly). An example output follows:

```
# time ./tctestprg4
real    0m5.08s
user    0m1.00s
sys     0m1.59s
```

The sum of user + sys is the total direct CPU cost of executing the program. This does not include the CPU costs of parts of the kernel that can be said to run on behalf of the program, but which do not actually run on the program's thread. For example, the cost of stealing page frames to replace the page frames taken from the free list when the program started is not reported as part of the program's CPU consumption. Another example of the **time** command is provided in 7.1, "CPU performance scenario" on page 250.

When starting to analyze a performance problem, most analysts start with the **vmstat** command, because it provides a brief overall picture of both CPU and memory usage.

2.3.1 The vmstat command

The **vmstat** command reports statistics about kernel threads, virtual memory, disks, traps, and CPU activity. Reports generated by the **vmstat** command can be used to balance system load activity. These system-wide statistics (among all processors) are calculated as averages for values expressed as percentages, and as sums otherwise. Most interesting from a CPU point of view are the highlighted two left-hand columns and the highlighted four right-hand columns in the following output:

```
# vmstat 2 4
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
```

```
 1  1 63572 173221    0    0    0    0    0    0 475 2625 401   0   0 99   0
 0  0 63579 173214    0    0    0    0    0    0 480 3279 406   0   1 98   0
 0  0 63579 173214    0    0    0    0    0    0 473 3157 386   0   0 99   0
 0  0 63579 173214    0    0    0    0    0    0 474 3158 398   0   1 99   0
```

The r column shows threads in the R state, while the b column shows threads in S state, as shown in Figure 2-2 on page 11. The four right-hand columns are a breakdown in percentages of CPU time used on user threads, system threads, CPU idle time (running the wait process), and CPU idle time when the system had outstanding disk or NFS I/O requests. For further discussion on the **vmstat** command, see 3.2, “The vmstat command” on page 53.

If a system has poor performance because of a lot of threads on the run queue or many threads waiting for I/O, then the **ps** command output is useful to determine which process has used the most CPU resources.

2.3.2 The ps command

The **ps** command is a flexible tool for identifying the programs that are running on the system and the resources they are using. It displays statistics and status information about processes on the system, such as process or thread ID, I/O activity, CPU, and memory utilization. In 3.3, “The ps command” on page 61, the **ps** command output relevant to a CPU tuning perspective is discussed.

When looking for a run-away process, the next step in the analysis is to find out which part of the process uses the CPU. For this, a profiler is needed. The AIX profiler of preference is **tprof**.

2.3.3 The tprof command

The **tprof** command can be run over a time period to trace the activity of the CPU. The CPU utilization is divided into kernel, user, shared, and other to show how many clock timer ticks were spent in each respective address space. If the user column shows high values, application tuning may be necessary. More information about the **tprof** command can be found in 3.4, “The tprof command” on page 67.

When finding a process that cannot be optimized, another way to tune the process is to lessen its priority in the run queue. This can be accomplished by grouping processes together to be handled by AIX Version 4.3 Workload Manager or by use of the **nice** and **renice** commands.

2.3.4 The nice and renice commands

The **nice** command can run a process at a priority lower than the process' normal priority. You must have root user authority to run a process at a higher priority. The priority of a process is often called its **nice** value, but while the priority of a process is recalculated at every clock timer tick, the nice value is stable and manipulated with the **nice** or **renice** commands. The nice value can range from 0 to 39, with 39 being the lowest priority. For example, if a process normally runs with a default nice value of 20, resetting the nice value with an increment of 5 runs the process at a lower priority, 25, and the process may run slower. More information about the priorities and nice values can be found in 6.1.1, "Priority calculation on AIX versions prior to 4.3.2" on page 207, 6.1.2, "Priority calculation on AIX Version 4.3.2 and later" on page 210, and 6.3.2, "The nice and renice commands" on page 218.

Finally, in the list of common performance tools, there is the **schedtune** command. This command is mentioned last for a reason: Do not manipulate the scheduler without thorough knowledge of the scheduler mechanism.

2.3.5 The schedtune command

The priority of most user processes varies with the amount of CPU time the process has used recently. The CPU scheduler's priority calculations are based on two variables, SCHED_R (the weighting factor) and SCHED_D (the decay factor). More information about the scheduler and the **schedtune** command is covered in 6.1, "The AIX scheduler" on page 206, and in 6.3.1, "The schedtune command" on page 214.

2.4 Memory performance overview

Memory in AIX is handled by the Virtual Memory Manager (VMM). The Virtual Memory Manager is a facility that makes real memory appear larger than its physical size. The virtual memory system is composed of real memory plus physical disk space where portions of memory that are not currently in use are stored.

The physical part of the virtual memory is divided into three types of segments that reflect where the data is stored. This is symbolized in Figure 2-3 on page 16.

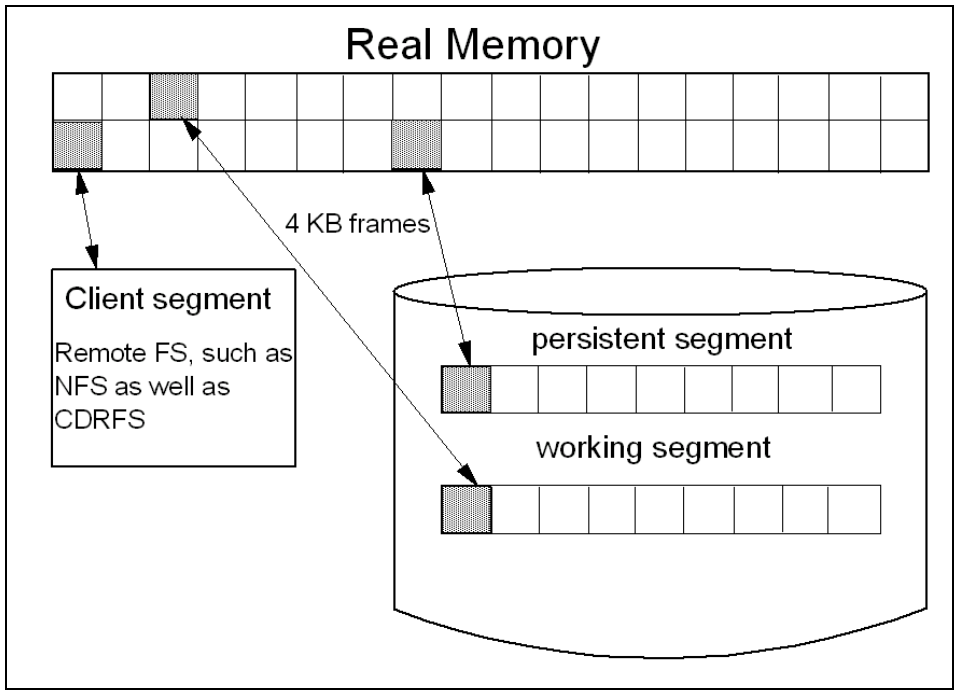


Figure 2-3 VMM segments from a client perspective

The three types of segments are as follows:

- Persistent segment

  A persistent segment persists after use by a process and has (and uses) permanent storage locations on disks. Files containing data or executable programs are mapped to persistent segments. AIX accesses all files as mapped files. This means that programs or file access are started with only a few initial disk pages, which are copied into virtual storage segments. Further pages are page-faulted in on demand.

- Working segment

  A working segment is transitory and only exists during use by the owning process. It has no permanent disk storage location and therefore is stored to paging space if free page frames in real memory are needed. For example, kernel text segments and the process stack are mapped onto working segments.

- Client segment

  A client segment is where the pages are brought in by CDRFS, NFS, or any other remote file system.

A process can use all of these segments, but from a process perspective the VMM is logically divided into:

- *Code* and *data* segments

  The code segment is the executable. This could be placed in a persistent (local) or a client (remote executable) segment. The data segment is data needed for the execution, for example, the process environment.

- *Private* and *shared* segments

  The private segment can be a working segment containing data for the process, for example, global variables, allocated memory, and the stack. Segments can also be shared among processes, for example, processes can share code segments, yet have private data segments.

The relationship between the segments is shown in Figure 2-4.

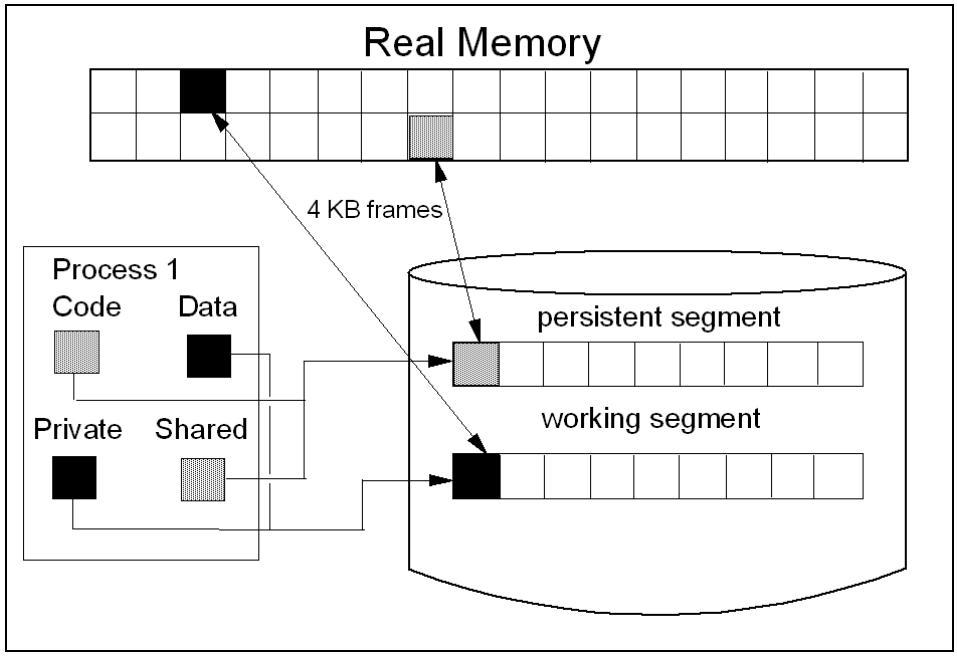


Figure 2-4 VMM segments from a process perspective

From a process point of view, the memory is further divided into 16 segments, each pointed to by a *segment register*. These segment registers are hardware registers located on the processor. When a process is active, the registers contain the addresses of the 16 segments addressable by that process. Each segment contains a specific set of information. A generic example is shown in Figure 2-5 on page 18.

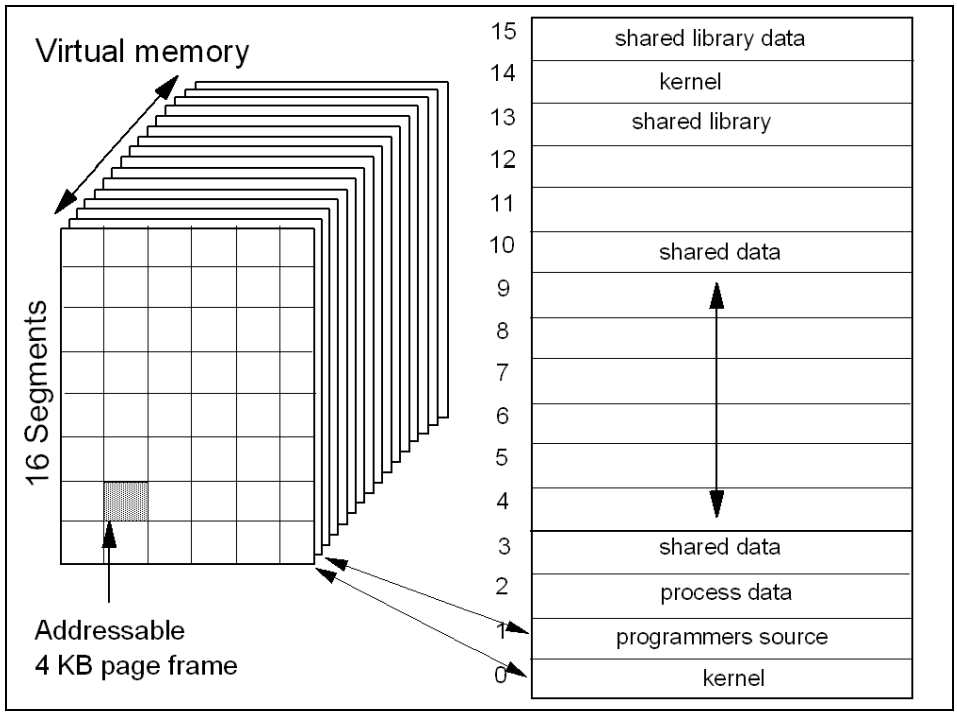


Figure 2-5 VMM memory registers

Each segment is further divided into 4069-byte pages of information. Each page sits on a 4 KB partition of the disk known as a slot. The VMM is responsible for allocating real memory page frames and resolving references to pages that are not currently in memory. In other words, when the system needs to reference a page that is not currently in memory, the VMM is responsible for finding and resolving the reference of the disk frame.

The VMM maintains a list of free page frames that is used to accommodate pages that must be brought into memory. In memory-constrained environments, the VMM must occasionally replenish the free list by moving some of the current data from real memory. This is called *page stealing*. A *page fault* is a request to load a 4-KB data page from disk. A number of places are searched in order to find data.

First, the data and instruction caches are searched. Next, the Translation Lookaside Buffer (TLB) is searched. This is an index of recently used virtual addresses with their page frame IDs. If the data is not in the TLB, the Page Frame Table (PTF) is consulted. This is an index for all real memory pages, and this index is held in pinned memory. The table is large; therefore, there are

indexes to this index. The Hash Anchor Table (HAT) links pages of related segments in order to get a faster entry point to the main PTF.

To the page stealer, memory is divided into computational memory and file memory.

- Computational memory consists of pages that belong to the working segment or program text segment.
- File memory consists of the remaining pages. These are usually pages from the permanent data file in persistent memory.

The page stealer tries to balance these two types of memory usage when stealing pages. The page replacement algorithm can be manipulated.

When starting a process, a slot is assigned, and when a process references a virtual memory page that is on the disk, the referenced page must be paged in and probably one or more pages must be paged out, creating I/O traffic and delaying the start up of the process. AIX attempts to steal real memory pages that are unlikely to be referenced in the near future, using a page replacement algorithm. If the system has too little memory, no RAM pages are good candidates to be paged out, as they will be reused in the near future. When this happens, continuous page in and page out occurs. This condition is called thrashing.

When discussing memory, the allocation algorithm is commonly mentioned. The following is a discussion from *AIX Version 4.3 System Management Concepts: Operating System and Devices*, SC23-4311, on the allocation algorithm:

> The operating system uses the PSALLOC environment variable to determine the mechanism used for memory and paging space allocation. If the PSALLOC environment variable is not set, is set to null, or is set to any value other than early, the system uses the default late allocation algorithm.
>
> The late allocation algorithm does not reserve paging space when a memory request is made; it approves the request and assigns paging space when pages are touched. Some programs allocate large amounts of virtual memory and then use only a fraction of the memory. Examples of such programs are technical applications that use sparse vectors or matrices as data structures. The late allocation algorithm is also more efficient for a real-time, demand-paged kernel such as the one in the operating system.
>
> For AIX Version 4.3.2 and later, the late allocation algorithm is modified to further delay the allocation of paging space. As mentioned previously, before AIX Version 4.3.2, paging space was allocated when a page was touched. However, this paging space may never be used, especially on systems with large real memory where paging is rare. Therefore, the allocation of paging space is delayed until it is necessary to page out the page, which results in no

wasted paging space allocation. This does result, however, in additional over-commitment of paging space. On a system where enough virtual memory is accessed that paging is necessary, the amount of paging space required may be as much as was required on previous releases.

It is possible to overcommit resources when using the late allocation algorithm for paging space allocation. In this case, when one process gets the resource before another, a failure results. The operating system attempts to avoid complete system failure by killing processes affected by the resource over-commitment. The SIGDANGER signal is sent to notify processes that the amount of free paging space is low. If the paging space situation reaches an even more critical state, selected processes that did not receive the SIGDANGER signal are sent a SIGKILL signal.

The user can use the PSALLOC environment variable to switch to an early allocation algorithm for memory and paging space allocation. The early allocation mechanism allocates paging space for the executing process at the time the memory is requested. If there is insufficient paging space available at the time of the request, the early allocation mechanism fails the memory request.

The new paging space allocation algorithm introduced with AIX Version 4.3.2 is also named Deferred Page Space Allocation (DPSA). After a page has been paged out to paging space, the disk block is reserved for that page if that page is paged back into RAM. Therefore, the paging space percentage-used value may not necessarily reflect the number of pages only in the paging space, because some of them may be back in RAM. If the page that was paged back in is the working storage of a thread, and if the thread releases the memory associated with that page or if the thread exits, then the disk block for that page is released. This affects the output for the **ps** command and the **svmon** command on Version 4.3.3. For more information on the differences between AIX Version 4.3.2 and Version 4.3.3 refer to *AIX Version 4.3 System Management Concepts: Operating System and Devices*, SC23-4311.

When working with memory performance tuning, the first command to use is usually the **vmstat** command.

2.4.1 The vmstat command

The **vmstat** command summarizes the total active virtual memory used by all of the processes running on the system, as well as the number of real-memory page frames on the free list. Active virtual memory is defined as the number of virtual-memory working segment pages that actually have been touched. This number can be larger than the number of real page frames in the machine because some of the active virtual-memory pages may have been written out to paging space.

When determining if a system is short on memory or if some memory tuning is required, use the **vmstat** command over a set interval and examine the pi and po columns on the resulting report. These columns indicate the number of paging space page-ins per second and the number of paging space page-outs per second. If the values are constantly non-zero, there may be a memory bottleneck. Having occasional non-zero values is not a concern, because paging is the main principle of virtual memory.

From a VMM tuning perspective, the middle (highlighted) columns are the most interesting. They provide information about the use of virtual and real memory and information about page faults and paging activity.

```
# vmstat 2 4
 kthr    memory             page                          faults          cpu
 -----  -----------  ------------------------     ------------   -----------
 r  b   avm    fre    re   pi   po   fr   sr   cy   in   sy    cs  us sy id wa
 0  0   16590  14475  0    0    0    0    0    0    101  9     8   50 0  50 0
 0  1   16590  14474  0    0    0    0    0    0    408  2232  48  0  0  99 0
 0  1   16590  14474  0    0    0    0    0    0    406  43    40  0  0  99 0
 0  1   16590  14474  0    0    0    0    0    0    405  91    39  0  0  99 0
```

The highlighted columns are described Table 2-3.

Table 2-3 VMM-related output from the vmstat command

Column	Description
avm	Active virtual pages
fre	Size of the free list
re	Pager input/output list
pi	Pages paged in from paging space
po	Pages paged out to paging space
fr	Pages freed (page replacement)
sr	Pages scanned by page-replacement algorithm
cy	Clock cycles by page-replacement algorithm

For more information about the **vmstat** command, see 3.2, “The vmstat command” on page 53.

Note: A large portion of real memory is utilized as a cache for file system data. It is not unusual for the size of the free list to remain small.

Another tool used in the initial phase of VMM tuning is the **ps** command.

2.4.2 The ps command

The **ps** command can also be used to monitor the memory usage of individual processes. The **ps v PID** command provides the most comprehensive report on memory-related statistics for an individual process, as discussed in 3.3, “The ps command” on page 61.

In the previous discussion, the paging space function of VMM was mentioned. The **lsps** command is an useful tool to check paging-space utilization.

2.4.3 The lsps command

The **lsps** command displays the characteristics of paging spaces, such as the paging-space name, physical-volume name, volume-group name, size, percentage of the paging space used, whether the space is active or inactive, and whether the paging space is set to be automatically initiated at system boot. The following is an example of the **lsps** command using the -a flag. The -s flag is useful when a summary and total percentage used over several disks is required.

```
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active Auto Type
hd6         hdisk2            rootvg        1024MB       1     yes   yes    lv
```

When finding problems with memory usage, the **svmon** command provides a more detailed report on what processes are using what segments of memory.

2.4.4 The svmon command

The **svmon** command provides a more in-depth analysis of memory usage. It is more informative, but also more intrusive, than the **vmstat** and **ps** commands. The **svmon** command captures a snapshot of the current state of memory. There are some significant changes in the flags and in the output from the **svmon** command between AIX Version 4.3.2 and Version 4.3.3. This is discussed in more detail in 3.5, “The svmon command” on page 70.

The command to use when tuning memory management is the **vmtune** command.

2.4.5 The vmtune command

The memory management algorithm tries to keep the size of the free list and the percentage of real memory occupied by persistent segment pages within specified bounds. These bounds can be altered with the **vmtune** command, which

can only be run by the root user. Changes made by this tool remain in effect until the next system reboot. More information on the **vmtune** command can be found in 6.5, “The vmtune command” on page 222.

To test how much (or, perhaps, little) memory is needed for a certain server load, use the **rmss** command.

2.4.6 The rmss command

The **rmss** command simulates a system with various sizes of real memory, without having to extract and replace memory boards. By running an application at several memory sizes and collecting performance statistics, you can determine the memory needed to run an application with acceptable performance. The **rmss** command can be invoked for the following purposes.

- To change the memory size and then exit. This lets you experiment freely with a given memory size.
- To function as a driver program. In this mode, the **rmss** command executes a specified command multiple times over a range of memory sizes, and displays important statistics describing command performance at each memory size. The command can be an executable or shell script file, with or without command line arguments.

2.5 Disk I/O performance overview

The set of operating system commands, library subroutines, and other tools that allow you to establish and control logical volume storage is called the Logical Volume Manager (LVM). The LVM controls disk resources by mapping data between simple and flexible logical views of storage space and the physical disks. The LVM does this using a layer of device driver code that runs above traditional disk device drivers.

The LVM consists of the logical volume device driver (LVDD) and the LVM subroutine interface library. The logical volume device driver is a pseudo-device driver that manages and processes all I/Os. It translates logical addresses into physical addresses and sends I/O requests to specific device drivers. When a process requests a disk read or write, the operation involves the file system, VMM, and LVM, as shown in Figure 2-6 on page 24.

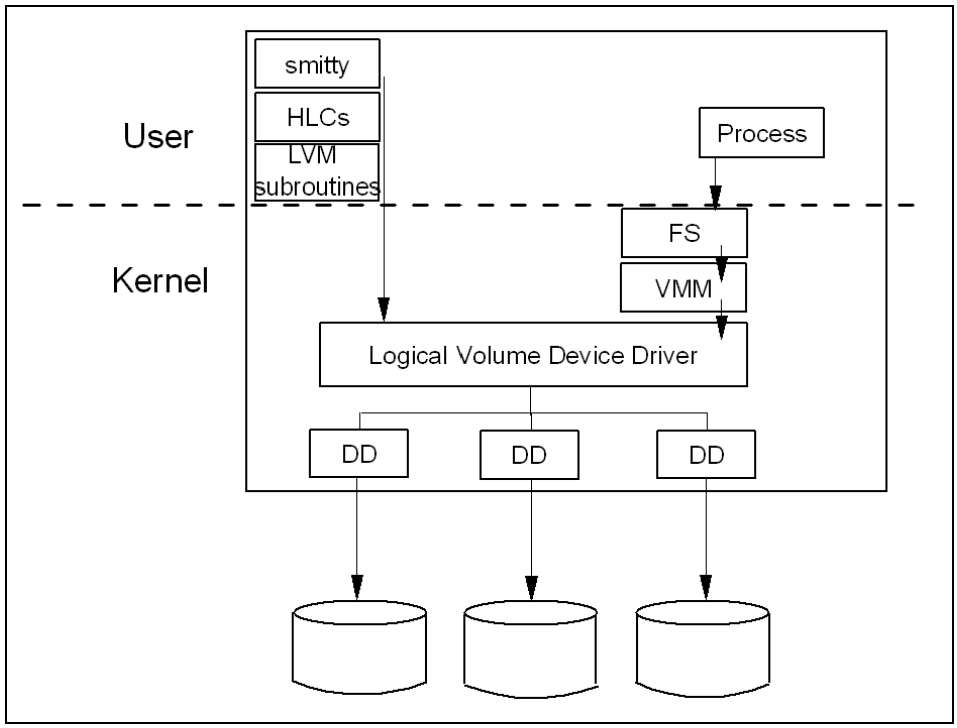


Figure 2-6 Logical volume device driver

Each individual disk drive, called a physical volume (PV), is named, such as /dev/hdisk0. If the physical volume is in use, it belongs to a volume group (VG). All of the physical volumes in a volume group are divided into physical partitions (PPs) of the same size (by default, 4 MB in volume groups that include physical volumes smaller than 4 GB; 8 MB or more with larger disks).

Within each volume group, one or more logical volumes (LVs) are defined. Each logical volume consists of one or more logical partitions. Each logical partition corresponds to at least one physical partition. If mirroring is specified for the logical volume, additional physical partitions are allocated to store the additional copies of each logical partition. Although the logical partitions are numbered consecutively, the underlying physical partitions are not necessarily consecutive or contiguous.

Figure 2-7 on page 25 shows the relationship and dependencies between the logical picture of the volume group with its corresponding physical layout.

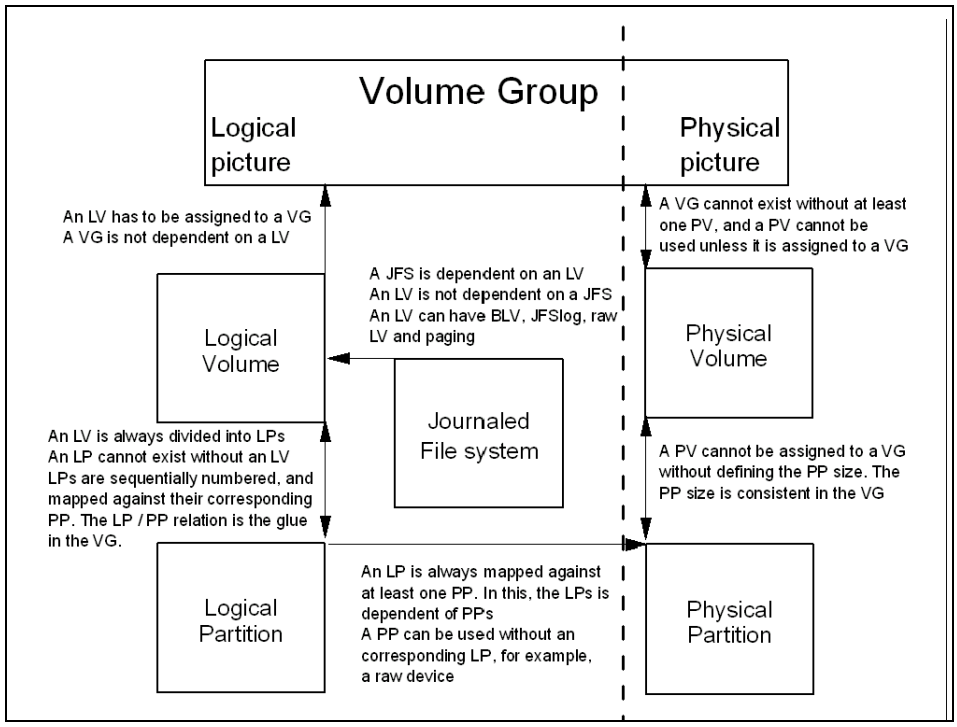


Figure 2-7 Dependencies in a volume group

Logical volumes can serve a number of system purposes, such as paging, but each logical volume that holds ordinary system data, user data, or programs contains a single journaled file system (JFS). Each JFS consists of a pool of page-size (4096-byte) blocks. When data is written to a file, one or more additional blocks are allocated to that file. These blocks may or may not be contiguous with one another and with other blocks previously allocated to the file.

In AIX Version 4, a given file system can be defined as having a fragment size of less than 4096 bytes. A fragment size can be set to 512, 1024, or 2048 bytes, allowing small files to be stored more efficiently.

While an operating system's file is conceptually a sequential and contiguous string of bytes, the physical reality is very different. Fragmentation may arise from multiple extensions to logical volumes, as well as allocation/release/reallocation activity within a file system. A file system is fragmented when its available space consists of large numbers of small chunks of space, making it impossible to write out a new file in contiguous blocks.

Access to files in a highly fragmented file system may result in a large number of seeks and longer I/O response times (seek latency dominates I/O response

time). For example, if the file is accessed sequentially, a file placement that consists of many widely separated chunks requires more seeks than a placement that consists of one or a few large contiguous chunks.

If the file is accessed randomly, a placement that is widely dispersed requires longer seeks than a placement where the file's blocks are close together.

The VMM tries to anticipate the future need for pages of a sequential file by observing the pattern a program uses to access the file. When the program accesses two successive pages of the file, the VMM assumes that the program will continue to access the file sequentially, and the VMM schedules additional sequential reads of the file. This is called *Sequential-Access Read Ahead*. These reads are overlapped with the program processing, and will make the data available to the program sooner than if the VMM had waited for the program to access the next page before initiating the I/O. The number of pages to be read ahead is determined by two VMM thresholds:

minpgahead A number of pages read ahead when the VMM first detects the sequential access pattern. If the program continues to access the file sequentially, the next read ahead will be for 2 x minpgahead, the next for 4 x minpgahead, and so on until the number of pages reaches maxpgahead.

maxpgahead A maximum number of pages the VMM will read ahead in a sequential file.

If the program deviates from the sequential-access pattern and accesses a page of the file out of order, sequential read ahead is terminated. It will be resumed with minpgahead pages if the VMM detects a resumption of sequential access by the program. The values of minpgahead and maxpgahead can be set with the **vmtune** command. More information on the **vmtune** command can be found in 6.5, “The vmtune command” on page 222.

To increase write performance, limit the number of dirty file pages in memory, reduce system overhead, and minimize disk fragmentation, the file system divides each file into 16 KB partitions. The pages of a given partition are not written to disk until the program writes the first byte of the next 16 KB partition. At that point, the file system forces the four dirty pages of the first partition to be written to disk. The pages of data remain in memory until their frames are reused. If a program accesses any of the pages before their frames are reused, no I/O is required.

If a large number of dirty file pages remains in memory and do not get reused, the sync daemon writes them to disk, which may result in abnormal disk utilization. To distribute the I/O activity more efficiently across the workload, *write-behind* can be turned on to tell the system how many pages to keep in

memory before writing them to disk. The write-behind threshold is on a per-file basis, which causes pages to be written to disk before the sync daemon runs. The I/O is spread more evenly throughout the workload.

There are two types of write-behind: Sequential and random. The size of the write-behind partitions and the write-behind threshold can be changed with the **vmtune** command.

Normal files are automatically mapped to segments to provide mapped files. This means that normal file access bypasses traditional kernel buffers and block I/O routines, allowing files to use more memory when the extra memory is available (file caching is not limited to the declared kernel buffer area).

Because most writes are asynchronous, FIFO I/O queues of several megabytes can build up, which can take several seconds to complete. The performance of an interactive process is severely impacted if every disk read spends several seconds working its way through the queue. In response to this problem, the VMM has an option called *I/O pacing* to control writes.

I/O pacing does not change the interface or processing logic of I/O. It simply limits the number of I/Os that can be outstanding against a file. When a process tries to exceed that limit, it is suspended until enough outstanding requests have been processed to reach a lower threshold.

Disk-I/O pacing is intended to prevent programs that generate very large amounts of output from saturating the system's I/O facilities and causing the response times of less-demanding programs to deteriorate. Disk-I/O pacing enforces per-segment (which effectively means per-file) *high* and *low water marks* on the sum of all pending I/Os. When a process tries to write to a file that already has high water mark pending writes, the process is put to sleep until enough I/Os have completed to make the number of pending writes less than or equal to the low water mark. The logic of I/O-request handling does not change. The output from high-volume processes is slowed down somewhat.

When gathering information on I/O performance, the first command to use is the **iostat** command.

2.5.1 The iostat command

The **iostat** command is used for monitoring system input/output device loading by observing the time the physical disks are active in relation to their average transfer rates. This command generates reports that can be used to change the system configuration to better balance the input/output load between physical disks and adapters. The **iostat** command gathers its information on the protocol layer.

AIX Version 4.3.3 and later contain enhancements to the method used to compute the percentage of CPU time spent waiting on disk I/O (wio time). The method used in AIX Version 4.3.2 and earlier versions of the operating system can, under certain circumstances, give an inflated view of wio time on SMPs. The wio time is reported by the commands **sar** (%wio), **vmstat** (wa), and **iostat** (% iowait).

In AIX Version 4.3.2 and earlier, at each clock timer tick interrupt on each processor (100 times a second per processor), a determination is made as to which of the four categories (usr/sys/wio/idle) to place the last 10 ms of time. If the CPU was busy in usr mode at the time of the clock interrupt, then usr gets the clock tick added into its category. If the CPU was busy in kernel mode at the time of the clock interrupt, then the sys category gets the tick. If the CPU was not busy, a check is made to see if any I/O-to-disk is in progress. If it is in progress, the wio category is incremented. If no disk I/O is in progress and the CPU is not busy, the idle category gets the tick.

The inflated view of wio time results from all idle CPUs being categorized as wio regardless of the number of threads waiting on I/O. For example, systems with just one thread doing I/O could report over 90 percent wio time, regardless of the number of CPUs it has.

The change in AIX Version 4.3.3 is to mark only an idle CPU as wio if an outstanding I/O was started on that CPU. This method can report much lower wio times when just a few threads are doing I/O and the system is otherwise idle. For example, a system with four CPUs and one thread doing I/O will report a maximum of 25 percent wio time. A system with 12 CPUs and one thread doing I/O will report a maximum of 8.3 percent wio time.

Also, NFS now goes through the buffer cache and waits in those routines are accounted for in the wa statistics.

Another change is that the wa column details the percentage of time the CPU was idle with pending disk I/O to not only local, but also NFS-mounted disks.

More information about the **iostat** command can be found in 4.2, “The iostat command” on page 118.

When experiencing performance problems due to disk I/O, the next step is to find the file system causing the problem. This can be done with the **filemon** command.

2.5.2 The filemon command

The **filemon** command uses the trace facility to obtain a detailed picture of I/O activity during a time interval on the various layers of file system utilization,

including the logical file system, virtual memory segments, LVM, and physical disk layers. Both summary and detailed reports are generated. More information about the **filemon** command can be found in 4.5.1, “The filemon command” on page 141.

If a file is identified as the problem, the **fileplace** command can be used to see how the file is stored.

2.5.3 The fileplace command

The **fileplace** command displays the placement of a specified file within the logical or physical volumes containing the file. By default, the **fileplace** command lists, to standard output, the ranges of logical volume fragments allocated to the specified file. More information about the **fileplace** command can be found in 4.6.2, “Enhanced journaled file system (JFS2)” on page 152.

If a logical volume is identified as a problem, the **lslv** command can provide useful information.

2.5.4 The lslv command

The **lslv** command shows, among other information, the logical volume fragmentation. If the workload shows a significant degree of I/O dependency, you can use the **lslv** command to investigate the physical placement of the files on the disk to determine if reorganization at some level would yield an improvement. More information about the **lslv** command can be found in 4.4.2, “Logical volume fragmentation” on page 138, and 4.9.3, “The lslv command” on page 165.

2.6 Network performance overview

When performance problems arise and you look for the cause, your local system may not have a problem, while the real problem is buildings away. An easy way to tell if the network is affecting overall performance is to compare those operations that involve the network with those that do not. If you are running a program that does a considerable amount of remote reads and writes and it is running slowly, but everything else seems to be running normally, then it is probably a network problem. Some of the potential network bottlenecks can be caused by the following:

- Client-network interface
- Network bandwidth
- Network topology

- Server network interface
- Server CPU load
- Server memory usage
- Server bandwidth
- Inefficient configuration

A large part of network tuning involves tuning TCP/IP to achieve maximum throughput. With the high-bandwidth interfaces such as FIDDI and Gigabit Ethernet, this has become even more important. Before attempting to tune network parameters, it helps to understand their use in the processing layer they affect. Figure 2-8 shows the layers involved in a read or write activity across TCP/IP and provides the network parameters used in each layer.

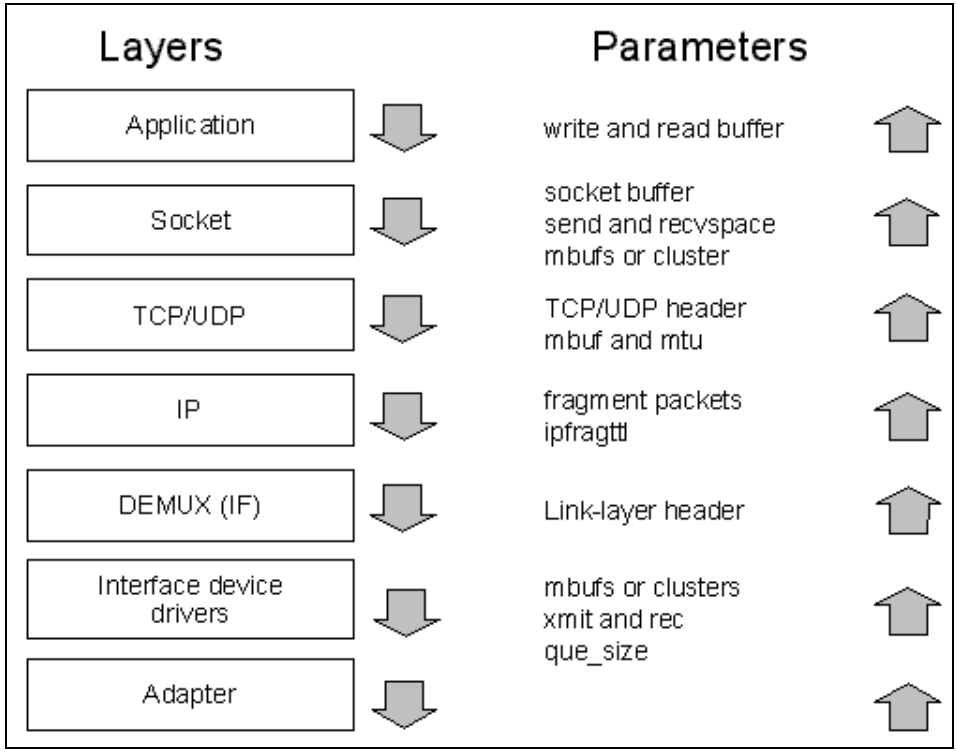


Figure 2-8 Network parameters

Chapter 5, “Network performance tools” on page 171, discusses, in more detail, how to set mbufs and clusters, network packet settings, and adapter settings for network performance tuning.

The first command to use for gathering information on network performance is the **netstat** command.

2.6.1 The netstat command

The **netstat** command symbolically displays the contents of various network-related data structures for active connections. The **netstat** command can also provide useful information on a per-protocol basis. For more information on the **netstat** command, see 5.4.3, “The netstat command” on page 180.

If the performance problem is due to NFS load, the **nfsstat** command is useful.

2.6.2 The nfsstat command

NFS gathers statistics on types of NFS operations performed, along with error information and performance indicators. You can use the **nfsstat** command to identify network problems and observe the type of NFS operations taking place on your system. The **nfsstat** command displays statistical information about the NFS and Remote Procedure Call (RPC) interfaces to the kernel. The **nfsstat** command splits its information into server and client parts. For example, use the command:

- **netstat -r** to see how an application uses NFS

  The output is divided into server connection- and connectionless-oriented, as well as client connection- and connectionless-oriented.

- **nfsstat -s** to see the server report

  The NFS server displays the number of NFS calls received (calls) and rejected (badcalls) due to authentication, as well as the counts and percentages for the various kinds of calls made.

- **nfsstat -c** to see the client part

  The NFS client displays the number of calls sent and rejected, as well as the number of times a client handle was received (clgets) and a count of the various kinds of calls and their respective percentages. For performance monitoring, the **nfsstat -c** command provides information on whether the network is dropping UDP packets. A network may drop a packet if it cannot handle it. Dropped packets can be the result of the response time of the network hardware or software or an overloaded CPU on the server. Dropped packets are not actually lost, because a replacement request is issued for them.

  A high badxid count implies that requests are reaching the various NFS servers, but the servers are too loaded to send replies before the client's RPC calls time out and are retransmitted. The badxid value is incremented each

time a duplicate reply is received for a transmitted request (an RPC request retains its XID through all transmission cycles). Excessive retransmissions place an additional strain on the server, further degrading response time.

The retrans column displays the number of times requests were retransmitted due to a time-out while waiting for a response. This situation is related to dropped UDP packets. If the retrans number consistently exceeds five percent of the total calls in column one, it indicates a problem with the server keeping up with demand.

For example, to display the number of RPC and NFS call-related information for the client and server, enter:

```
# nfsstat -rs

Server rpc:
Connection oriented
calls      badcalls   nullrecv   badlen     xdrcall    dupchecks  dupreqs
0          0          0          0          0          0          0
Connectionless
calls      badcalls   nullrecv   badlen     xdrcall    dupchecks  dupreqs
0          0          0          0          0          0          0
```

RPC output for the server is as follows:

calls Total number of RPC calls received from clients.

badcalls Total number of calls rejected by the RPC layer.

nullrecv Number of times an RPC call was not available when it was thought to be received. If the number starts to grow, it may mean there are too many nfsd daemons.

badlen Packets truncated or damaged (number of RPC calls with a length shorter than a minimum-sized RPC call).

xdrcall Number of RPC calls whose header could not be External Data Representation (XDR) decoded.

dupchecks Number of RPC calls looked up in the duplicate request cache.

dupreqs Number of duplicate RPC calls found.

When requiring a more detailed output, the **netpmon** command, a trace facility, is useful.

2.6.3 The netpmon command

The **netpmon** command monitors a trace of system events, and reports on network activity and performance during the monitored interval. By default, the

netpmon command runs in the background while one or more application programs or system commands are being executed and monitored. The **netpmon** command automatically starts and monitors a trace of network-related system events in real time. More information on the **netpmon** command can be found in Chapter 5, “Network performance tools” on page 171.

2.7 The performance diagnostic tool (PDT)

The performance diagnostic tool attempts to identify performance problems automatically by collecting and integrating a wide range of performance, configuration, and availability data. The data is regularly evaluated to identify and anticipate common performance problems.

PDT assesses the current state of a system and tracks changes in workload and performance. It attempts to identify incipient problems and suggests solutions before the problems become critical.

For the most part, PDT functions with no required user input. PDT data collection and reporting are easily enabled, and then no further administrator activity is required. Periodically, data is collected and recorded for historical analysis, and a report is produced and mailed to the adm user. Normally, only the most significant apparent problems are recorded on the report. If there are no significant problems, that fact is reported. PDT can be customized to direct its report to a different user or to report apparent problems of a lower severity level.

2.7.1 Installing and enabling PDT

PDT is installed through the **installp** command as the bos.perf.diag_tool fileset. PDT must be enabled in order to begin collecting data and writing reports. Enable PDT by executing the /usr/sbin/perf/diag_tool/pdt_config script. Only the root user is permitted to run this script.

The following example shows the message displayed when the /usr/sbin/perf/diag_tool/pdt_config script is run:

```
# /usr/sbin/perf/diag_tool/pdt_config
_________________PDT customization menu____________________

1) show current  PDT report recipient and severity level
2) modify/enable PDT reporting
3) disable       PDT reporting
4) modify/enable PDT collection
5) disable       PDT collection
6) de-install    PDT
7) exit pdt_config
```

```
Please enter a number:
```

When you respond with 4, default PDT collection and reporting is enabled. The crontab entry for user adm is updated to add the PDT entries. The entries execute a shell script called Driver_ in the /usr/sbin/perf/diag_tool directory. This script is passed to three different parameters, each representing a collection profile, at three different collection times. To terminate the pdt_config program, respond with 7. To disable collection, respond with 5.

Table 2-4 lists the performance diagnostic tool's configurable attributes.

Table 2-4 The performance diagnostic tool

PDT configurable attribute	PDT configuration method
PDT report recipient	Execute the /usr/sbin/perf/diag_tool/pdt_config script.
PDT security level	Execute the /usr/sbin/perf/diag_tool/pdt_config script.
PDT report on demand	Execute the /usr/sbin/perf/diag_tool/pdt_report script.
List of files monitored by PDT	Edit the /var/perf/cfg/diag_tool/.files file.
Modifying the list of hosts monitored by PDT	Edit the /var/perf/cfg/diag_tool/.nodes file.
Historical-record retention period for PDT reports	Edit the /var/perf/cfg/diag_tool/.retention.list.
PDT collection, retention, and reporting times	Use the /var/perf/cfg/diag_tool/.collection.control file. Use the /var/perf/cfg/diag_tool/.retention.control file. Use the /var/perf/cfg/diag_tool/.reporting.control file. The default time can be changed by altering the crontab for user adm.
PDT thresholds	Uses the /var/perf/cfg/diag_tool/.thresholds file.
PDT error reporting	View the /var/perf/tmp/.stderr file.
Uninstalling PDT	Execute the /usr/sbin/perf/diag_tool/pdt_config script Use installp to remove the bos.perf.diag_tool fileset.

2.8 Service level agreement

To allow the service provider and the client to manage the environment, a service level agreement should be created.

A service level agreement (SLA) is a contract between a service provider and a client that specifies, usually in measurable terms, what service the service provider will furnish. The service level agreement should include the following metrics in order to safeguard the service provider and the customer, for example:

- Reporting policies
- Roles and responsibilities
- What percentage of the time service will be available
- The number of users that can be served simultaneously
- Specific performance benchmarks to which actual performance will be periodically compared
- Goals and objectives
- The schedule for notification in advance of changes that may affect users
- Help desk response time for various classes of problems
- Penalties
- Dial-in access availability
- Usage statistics that will be provided
- Incentives

2.9 Summary

Figure 2-1 on page 10 (shown in the beginning of this chapter) is used in the summary (Figure 2-9 on page 36). It is modified to include performance suggestions.

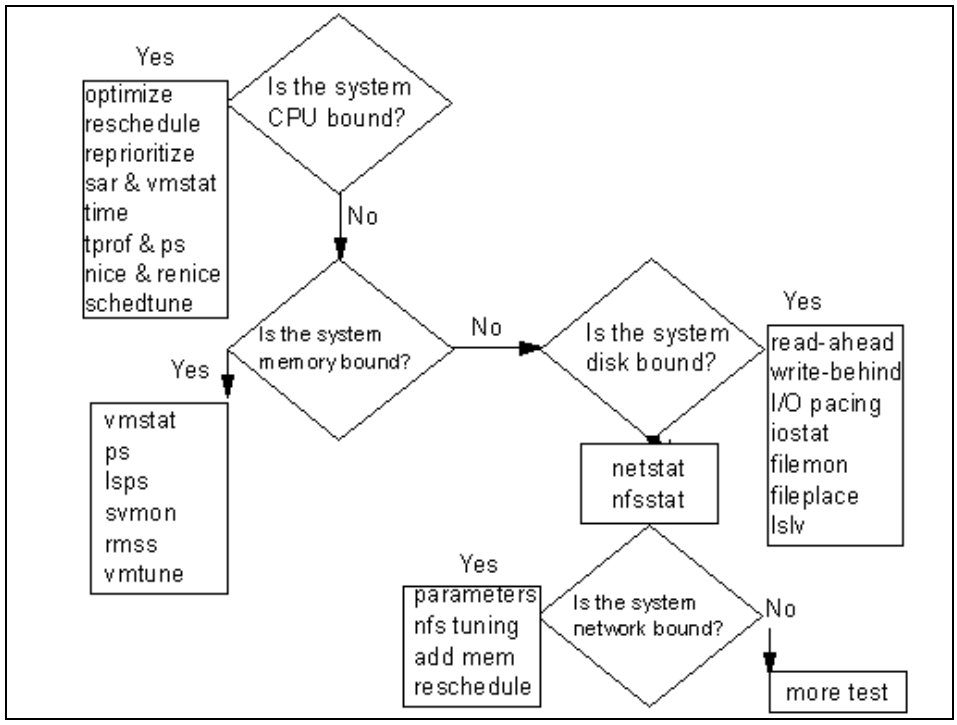


Figure 2-9 Performance tuning flowchart

2.10 Quiz

The following assessment question helps verify your understanding of the topics discussed in this chapter.

1. When paging, cache, and Translation Lookaside Buffer (TLB) metrics or statistics are analyzed for performance, which of the following resources is being analyzed?

   A. CPU

   B. Memory

   C. Disk I/O

   D. Network I/O

2.10.1 Answers

The following is the preferred answer to the question provided in this section.

1. B

3

CPU and memory performance monitoring tools

This chapter takes a detailed look at several CPU and memory performance monitoring tools. The provided examples will assist you in culling the valuable information from the verbose output generated by these tools.

3.1 The sar command

The **sar** command (System Activity Report) can be used in two ways to collect data: One is to view system data in real time, and the other is to view data previously captured.

The **sar** command is one of the first performance monitors to be used by an administrator. Although the **sar** command does give useful information on most system functions, it should be noted that there are other tools that will give more accurate system utilization reports on specific sections within the environment.

3.1.1 Accounting software

The **sar** command requires the bos.perf.perfstat default installation fileset and the bos.acct additional fileset to be installed. To view the files in bos.acct, type the following command:

```
# lslpp -f bos.acct
```

3.1.2 Examples of using the sar command

The **sar** command, without any flags, will give an output of every line in the file for the current day as collected by the **sa1** command. The time slice can be set up in the crontab file and. In the following example, it uses the default in the /var/spool/cron/crontabs/adm file:

```
# sar
08:00:00    %usr    %sys    %wio   %idle
08:20:00       0       0       0     100
08:40:00       0       0       0     100
09:00:00       0       0       0     100

Average        0       0       0     100
```

The **sar** command, using only interval and number flags, will have the output shown in the following example. This is the same as running the **sar -u 1 10** command. The 1 specifies the interval in seconds and the 10 specifies the number of times the data is captured.

```
# sar 1 10

AIX server2 3 4 000FA17D4C00    06/30/00

09:14:57    %usr    %sys    %wio   %idle
09:14:58      54      18      28       0
09:14:59      40      20      40       0
09:15:00      44      19      38       0
```

```
09:15:01      82       14        4        0
09:15:02      66       16       18        0
09:15:03      45       12       43        0
09:15:04      60       17       23        0
09:15:05      47       16       37        0
09:15:06      65       12       23        0
09:15:07      48        8       44        0

Average       55       15       30        0
```

The **sar -a** command reports the use of file access system routines specifying how many times per second several of the system file access routines have been called.

```
# sar -a 1 10

AIX server2 3 4 000FA17D4C00    06/30/00

09:28:44  iget/s lookuppn/s dirblk/s
09:28:45       0       1169      277
09:28:46       0         15        0
09:28:47       0         50        0
09:28:48       0        559       19
09:28:49       0        390       20
09:28:50       0       1467      137
09:28:51       0       1775      153
09:28:52       0       2303       74
09:28:53       0       2832       50
09:28:54       0        883       44

Average        0       1144       77
```

The **sar -c** command reports system calls.

```
# sar -c 1 10

AIX server2 3 4 000FA17D4C00    06/30/00

09:33:04 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
09:33:05    1050     279     118    0.00    0.00  911220 5376749
09:33:06     186      19      74    0.00    0.00    3272 3226417
09:33:07     221      19      79    0.00    0.00    3272 3277806
09:33:08    2996     132     400    0.00    0.00  314800 2284933
09:33:09    3304     237     294    0.00    0.00  167733  848174
09:33:10    4186     282     391    0.00    0.00  228196  509414
09:33:11    1938     109     182    1.00    1.00  153703 1297872
09:33:12    3263     179     303    0.00    0.00  242048 1003364
09:33:13    2751     172     258    0.00    0.00  155082  693801
09:33:14    2827     187     285    0.00    0.00  174059 1155239
```

```
Average     2273     162     238    0.10    0.10  235271 1966259
```

The **sar -d** command reads disk activity with read and write and block size averages. This flag is not documented in the AIX documentation, but is used by some AIX gurus. Use the **iostat** command instead of the **sar -d** command.

```
# sar -d 5 3

AIX server2 3 4 000FA17D4C00    06/30/00

10:08:19     device    %busy    avque    r+w/s   blks/s   avwait   avserv

10:08:24     hdisk0      0      0.0        0        4      0.0      0.0
             hdisk1      0      0.0        0        3      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0

10:08:29     hdisk0     44      1.0      366     3569      0.0      0.0
             hdisk1     36      0.0       47     2368      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0

10:08:34     hdisk0     84      2.0      250     1752      0.0      0.0
             hdisk1     16      1.0       19      950      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0

Average      hdisk0     42      1.0      205     1775      0.0      0.0
             hdisk1     17      0.3       22     1107      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0
```

The **sar -q** command reports queue statistics.

```
# sar -q 1 10

AIX server2 3 4 000FA17D4C00    06/30/00

11:08:33 runq-sz %runocc swpq-sz %swpocc
11:08:34                    1.0     100
11:08:35                    1.0     100
11:08:36                    1.0     100
11:08:37     1.0     100
11:08:38     1.0     100    1.0     100
11:08:39     1.0     100    1.0     100
11:08:40     1.0     100    1.0     100
11:08:41                    1.0     100
11:08:42                    1.0     100
11:08:43     1.0     100

Average      1.0      50    1.0      80
```

The **sar -r** command reports paging statistics.

```
# sar -r 1 10

AIX server2 3 4 000FA17D4C00    06/30/00

11:16:11   slots cycle/s fault/s  odio/s
11:16:12  130767    0.00  472.82   66.02
11:16:13  130767    0.00  989.00  800.00
11:16:14  130767    0.00   44.00 1052.00
11:16:15  130767    0.00   43.00 1040.00
11:16:16  130767    0.00   47.00 1080.00
11:16:17  130767    0.00   43.00  808.00
11:16:18  130767    0.00   40.00  860.00
11:16:19  130767    0.00   46.00  836.00
11:16:20  130767    0.00   47.00  852.00
11:16:21  130767    0.00   48.00  836.00

Average   130767       0     183     821
```

The **sar -v** command reports the status of the process, kernel-thread, inode, and file tables.

```
# sar -v 1 5

AIX server2 3 4 000FA17D4C00    06/30/00

11:12:39  proc-sz     inod-sz     file-sz     thrd-sz
11:12:40  49/262144   229/42942   315/511     59/524288
11:12:41  46/262144   221/42942   303/511     56/524288
11:12:42  45/262144   220/42942   301/511     55/524288
11:12:43  45/262144   220/42942   301/511     55/524288
11:12:44  45/262144   220/42942   301/511     55/524288
```

The **sar -y** command reports TTY device activity per second.

```
# sar -y 1 10

AIX server2 3 4 000FA17D4C00    06/30/00

11:48:36 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
11:48:37       0       0     104      63      60       0
11:48:38       0       0      58       9      60       0
11:48:39       0       0      58      69      61       0
11:48:40       0       0      58      68      60       0
11:48:41       0       0      58      69       3       0
11:48:42       0       0      58      68      52       0
11:48:43       0       0      58      69      60       0
11:48:44       0       0      58      25      60       0
11:48:45       0       0      58      42      23       0
```

```
11:48:46       0       0      58      68       9       0

Average        0       0      63      55      45       0
```

Although this is not an exhaustive list of the **sar** command and command output, it is an indication of what the main flags can do. When running the **sar** command, a combination of the flags can be used to obtain the output required for analyses, as in the following example:

```
# sar -y -r 1 5
AIX server2 3 4 000FA17D4C00    06/30/00

11:48:56 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
           slots cycle/s fault/s  odio/s

11:48:57       0       0     147      67       3       0
          130767    0.00    3.96    0.00

11:48:58       0       0     102      69      58       0
          130767    0.00    0.00    0.00

11:48:59       0       0     102      68      60       0
          130767    0.00    0.00    0.00

11:49:00       0       0     102      69      17       0
          130767    0.00    0.00    0.00

11:49:01       0       0     102      68       3       0
          130767    0.00    1.00    4.00


Average        0       0     111      68      28       0
Average   130767       0       1       1
```

3.1.3 The sar command summary

The **sar** command writes the contents of selected cumulative activity counters in the operating system to standard output.

The **sar** command syntax is as follows:

```
sar [ { -A | [ -a ] [ -b ] [ -c ] [ -k ] [ -m ] [ -q ] [ -r ] [ -u ] [ -v ]
[ -w ] [ -y ] } ] [ -P ProcessorIdentifier, ... | ALL ]
[ -ehh [ :mm [ :ss ] ] ] [ -fFile ] [ -iSeconds ] [ -oFile ]
[ -shh [ :mm [ :ss ] ] ] [ interval [ number ] ]
```

The accounting system, based on the values in the interval and number parameters, writes information at the specified intervals in seconds the specified number of times. The default sampling interval for the number parameter is 1

second. The collected data can also be saved in the file specified by the -o file flag.

If CPU utilization is near 100 percent (user + system), the workload sampled is CPU-bound. If a considerable percentage of time is spent in I/O wait, it implies that CPU execution is blocked while waiting for disk I/O. The I/O may be from file accesses or it may be I/O associated with paging due to a lack of sufficient memory.

Note: The time the system spends waiting for remote file access is not accumulated in the I/O wait time. If CPU utilization and I/O wait time for a task are relatively low, and the response time is not satisfactory, consider investigating how much time is being spent waiting for remote I/O. Since no high-level command provides statistics on remote I/O wait, trace data may be useful in observing this.

The **sar** command calls a command named **sadc** to access system data. Two shell scripts, /usr/lib/sa/sa1 and /usr/lib/sa/sa2, are structured to be run by the **cron** command and provide daily statistics and reports. Sample stanzas are included (but commented out) in the /var/spool/cron/crontabs/adm crontab file to specify when the cron daemon should run the shell scripts. Collection of data in this manner is useful to characterize system usage over a period of time and determine peak usage hours.

The commonly used flags of the **sar** command are provided in Table 3-1.

Table 3-1 Commonly used flags of the sar command

Flag	Description
-A	Without the -P flag, this is equivalent to specifying -abckmqruvwy. With the -P flag, this is equivalent to specifying -acmuw.

Flag	Description
-a	Reports use of file access system routines, specifying how many times per second several of the system file access routines have been called. When used with the -P flag, the information is provided for each specified processor; otherwise, it is provided only system-wide. The following values are displayed: ► dirblk/s Number of 512-byte blocks read by the directory search routine to locate a directory entry for a specific file. ► iget/s Number of calls to any of several inode lookup routines that support multiple file system types. The iget routines return a pointer to the inode structure of a file or device. ► lookuppn/s Calls to the directory search routine that finds the address of a v-node, given a path name.

Flag	Description
-b	Reports buffer activity for transfers, accesses, and cache (kernel block buffer cache) hit ratios per second. Access to most files in AIX Version 3 bypasses kernel block buffering, and therefore does not generate these statistics. However, if a program opens a block device or a raw character device for I/O, traditional access mechanisms are used, making the generated statistics meaningful. The following values are displayed: ► bread/s, bwrit/s Reports the number of block I/O operations. These I/Os are generally performed by the kernel to manage the block buffer cache area, as discussed in the description of the lread/s value. ► lread/s, lwrit/s Reports the number of logical I/O requests. When a logical read or write to a block device is performed, a logical transfer size of less than a full block size may be requested. The system accesses the physical device units of complete blocks and buffers these blocks in the kernel buffers that have been set aside for this purpose (the block I/O cache area). This cache area is managed by the kernel, so that multiple logical reads and writes to the block device can access previously buffered data from the cache and require no real I/O to the device. Application read and write requests to the block device are reported statistically as logical reads and writes. The block I/O performed by the kernel to the block device in management of the cache area is reported as block reads and block writes. ► pread/s, pwrit/s Reports the number of I/O operations on raw devices. Requested I/O to raw character devices is not buffered as it is for block devices. The I/O is performed to the device directly. ► %rcache, %wcache Reports caching effectiveness (cache hit percentage). This percentage is calculated as: [(100) x (lreads - breads) / (lreads)].

Flag	Description
-c	Reports system calls. When used with the -P flag, the information is provided for each specified processor; otherwise, it is provided only system-wide. The following values are displayed: ► exec/s, fork/s Reports the total number of fork and exec system calls. ► sread/s, swrit/s Reports the total number of read/write system calls. ► rchar/s, wchar/s Reports the total number of characters transferred by read/write system calls ► scall/s Reports the total number of system calls.
-e *hh[:mm[:ss]]*	Sets the ending time of the report. The default ending time is 18:00.
-f *file*	Extracts records from file (created by -o *file* flag). The default value of the file parameter is the current daily data file (the /var/adm/sa/sadd file).
-i *seconds*	Selects data records at seconds as close as possible to the number specified by the seconds parameter. Otherwise, the **sar** command reports all seconds found in the data file.
-k	Reports kernel process activity. The following values are displayed: ► kexit/s Reports the number of kernel processes terminating per second ► kproc-ov/s Reports the number of times kernel processes could not be created because of enforcement of process threshold limit ► ksched/s Reports the number of kernel processes assigned to tasks per second.

Flag	Description
-m	Reports message (sending and receiving) and semaphore (creating, using, or destroying) activities per second. When used with the -P flag, the information is provided for each specified processor; otherwise, it is provided only system-wide. The following values are displayed: ► msg/s Reports the number of IPC message primitives ► sema/s Reports the number of IPC semaphore primitives
-o *File*	Saves the readings in the file in binary form. Each reading is in a separate record and each record contains a tag identifying the time of the reading.
-P *ProcessorIdentifier*, ... | ALL	Reports per-processor statistics for the specified processor or processors. Specifying the ALL keyword reports statistics for each individual processor, and globally for all processors. Of the flags that specify the statistics to be reported, only the -a, -c, -m, -u, and -w flags are meaningful with the -P flag.
-q	Reports queue statistics. The following values are displayed: ► runq-sz Reports the average number of kernel threads in the run queue ► %runocc Reports the percentage of the time the run queue is occupied ► swpq-sz Reports the average number of kernel threads waiting to be paged in ► %swpocc Reports the percentage of the time the swap queue is occupied

Flag	Description
-r	Reports paging statistics. The following values are displayed: ► cycle/s Reports the number of page replacement cycles per second. ► fault/s Reports the number of page faults per second. This is not a count of page faults that generate I/O, because some page faults can be resolved without I/O. ► slots Reports the number of free pages on the paging spaces. ► odio/s Reports the number of nonpaging disk I/Os per second.
-s *hh[:mm[:ss]]*	Sets the starting time of the data, causing the **sar** command to extract records time-tagged at, or following, the time specified. The default starting time is 08:00.
-u	Reports per processor or system-wide statistics. When used with the -P flag, the information is provided for each specified processor; otherwise, it is provided only system-wide. Because the -u flag information is expressed as percentages, the system-wide information is simply the average of each individual processor's statistics. Also, the I/O wait state is defined system-wide and not per processor. The following values are displayed: ► %idle Reports the percentage of time the CPU or CPUs were idle with no outstanding disk I/O requests. ► %sys Reports the percentage of time the CPU or CPUs spent in execution at the system (or kernel) level. ► %usr Reports the percentage of time the CPU or CPUs spent in execution at the user (or application) level. ► %wio Reports the percentage of time the CPU or CPUs were idle waiting for disk I/O to complete. For system-wide statistics, this value may be slightly inflated if several processors are idling at the same time (an unusual occurrence).

Flag	Description
-v	Reports status of the process, kernel-thread, inode, and file tables. The following values are displayed file-sz, inod-sz, proc-sz, thrd-sz. Reports the number of entries in use for each table.
-w	Reports system switching activity. When used with the -P flag, the information is provided for each specified processor; otherwise, it is provided only system-wide. The following value is displayed pswch/s. Reports the number of context switches per second.
-y	Reports tty device activity per second. ► canch/s Reports tty canonical input queue characters. This field is always 0 (zero) for AIX Version 4 and higher. ► mdmin/s Reports tty modem interrupts. ► outch/s Reports tty output queue characters. ► rawch/s Reports tty input queue characters. ► revin/s Reports tty receive interrupts. ► xmtin/s Reports tty transmit interrupts.

Note: The **sar** command:

- The **sar** command itself can generate a considerable number of reads and writes, depending on the interval at which it is run. Run the **sar** statistics without the workload to understand the **sar** command's contribution to your total statistics.
- The **sar** command reports system unit activity if no other specific content options are requested.

3.1.4 The sadc command

The **sadc** command provides a system data collector report.

The syntax for the **sadc** command is as follows:

```
sadc [ interval number ] [ outfile ]
```

It samples system data at a specified interval measured in seconds (*interval*) a specified number of times (*number*). It writes in binary format to the specified file (*outfile*) or to the standard output. When both *interval* and *number* are not specified, a dummy record, which is used at system startup to mark the time when the counter restarts from 0, will be written. The **sadc** command is intended to be used as a backend to the **sar** command.

AIX contains a number of counters that are increased as various system actions occur. The various system actions include:

- System unit utilization counters
- Buffer usage counters
- Disk and tape I/O activity counters
- TTY device activity counters
- Switching and subroutine counters
- File access counters
- Queue activity counters
- Interprocess communication counters

3.1.5 The sa1 and sa2 commands

The **sa1** command is a shell procedure variant of the **sadc** command and handles all of the flags and parameters of that command. The **sa1** command collects and stores binary data in the /var/adm/sa/sa*dd* file, where *dd* is the day of the month.

The syntax for the **sa1** command is as follows:

```
sa1 [ interval number ]
```

The interval and number parameters specify that the record should be written a number times at an interval of seconds. If you do not specify these parameters, a single record is written.

The **sa1** command is designed to be started automatically by the **cron** command. If the **sa1** command is not run daily from the **cron** command, the **sar** command displays a message about the nonexistence of the /usr/lib/sa/sa1 data file.

The **sa2** command is a variant shell procedure of the **sar** command, which writes a daily report in the /var/adm/sa/sar*dd* file, where *dd* is the day of the month. The **sa2** command handles all of the flags and parameters of the **sar** command.

The **sa2** command is designed to be run automatically by the **cron** command and run concurrently with the **sa1** command.

The syntax for the **sa2** command is as follows:

```
sa2
```

3.2 The vmstat command

The **vmstat** command reports statistics about kernel threads, virtual memory, disks, traps, and CPU activity. Reports generated by the **vmstat** command can be used to balance system load activity. These system-wide statistics (among all processors) are calculated as averages for values expressed as percentages, and as sums otherwise.

The **vmstat** command syntax is as follows:

```
vmstat [ -f ] [ -i ] [ -s ] [ PhysicalVolume ] [ interval [ count ] ]
```

If the **vmstat** command is invoked without flags, the report contains a summary of the virtual memory activity since system startup. If the -f flag is specified, the **vmstat** command reports the number of forks since system startup. The PhysicalVolume parameter specifies the name of the physical volume.

An example of the **vmstat** command without any flags follows:

```
# vmstat
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 15982  1388   0   0   0   8   22   0 113  281  36  1  0 98  1
```

Below is an example of the **vmstat** command with the -f flag:

```
# vmstat -f
      51881 forks
```

The interval parameter specifies the amount of time in seconds between each report. The first report contains statistics for the time since system startup. Subsequent reports contain statistics collected during the interval since the previous report. If the interval parameter is not specified, the **vmstat** command generates a single report and then exits. The count parameter can only be specified with the interval parameter. If the count parameter is specified, its value determines the number of reports generated and the frequency data is collected in seconds. If the interval parameter is specified without the count parameter, reports are continuously generated. A count parameter of 0 is not allowed.

The following is an example of the **vmstat** command with the interval and count parameters:

```
# vmstat 1 5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 15982  1388   0   0   0   8   22   0 113  281  36  1  0 98  1
 0  0 15982  1387   0   0   0   0    0   0 108 4194  31  2  3 95  0
 0  0 15982  1387   0   0   0   0    0   0 109  286  30  0  0 99  0
 0  0 15982  1387   0   0   0   0    0   0 108  285  26  0  0 99  0
 0  0 15982  1387   0   0   0   0    0   0 111  286  32  0  0 99  0
```

The kernel maintains statistics for kernel threads, paging, and interrupt activity. For disks, the average transfer rate is determined by using the active time and number of transfers information. The percent active time is computed from the amount of time the drive is busy during the report.

The **vmstat** command with additional information regarding a specific disk is as follows:

```
# vmstat hdisk1
kthr     memory             page              faults        cpu      disk xfer
----- ----------- ------------------------ ------------ ----------- -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa  1  2  3  4
 0  0 16273  8385   0   0   0   9   22   0 115  284  39  1  1 98  1  0
```

In the previous example of a report generated by the **vmstat** command, the column headings and their descriptions have the following meaning.

kthr Kernel thread state changes per second over the sampling interval.

- **r** Number of kernel threads placed in run queue
- **b** Number of kernel threads placed in wait queue (awaiting resource, awaiting input or output)

Memory Information about the usage of virtual and real memory. Virtual pages are considered active if they are allocated. A page is 4096 bytes.

- **avm** Active virtual pages. When a process executes, space for working storage is allocated on the paging devices (backing store). This can be used to calculate the amount of paging space assigned to executing processes. The number in the avm field divided by 256 will yield the number of megabytes (MB), system wide, allocated to page space. The **lsps -a** command also provides information on individual paging space. It is

recommended that enough paging space be configured on the system so that the paging space used does not approach 100 percent. When fewer than 128 unallocated pages remain on the paging devices, the system will begin to kill processes to free some paging space.

fre Size of the free list. The system maintains a buffer of memory frames, called the free list, that will be readily accessible when the VMM needs space. The nominal size of the free list varies depending on the amount of real memory installed. On systems with 64 MB of memory or more, the minimum value (MINFREE) is 120 frames. For systems with less than 64 MB, the value is two times the number of MB of real memory, minus 8. For example, a system with 32 MB would have a MINFREE value of 56 free frames. The MINFREE and MAXFREE limits can be shown using the **vmtune** command.

Note: A large portion of real memory is utilized as a cache for file system data. It is not unusual for the size of the free list to remain small.

Page Information about page faults and paging activity. These are averaged over the interval and given in units per second. The AIX VMM implements a technique called Early Allocation of Paging Space. When a page is allocated in RAM, and it is not a Client (NFS) or a Persistent (disk file) Storage Page, then it is considered a Working Storage Page. Working Storage Pages are commonly an application's stack, data, and any shared memory segments. So, when a program's stack or data area is increased, and RAM is accessed, the VMM will allocate space in RAM and space on the paging device. This means that even before RAM is exhausted, paging space is used.

re Pager input/output list. The number of reclaims per second. During a page fault, when the page is on the free list and has not been reassigned, this is considered a reclaim because no new I/O request has been initiated. It also includes the pages last requested by the VMM for which I/O has not been completed or those prefetched by VMM's read-ahead mechanism but hidden from the faulting segment.

pi Pages paged in from paging space.

po Pages paged out to paging space.

Note: pi and po values only include Working Storage Pages, not Persistent Storage Pages.

fr Pages freed (page replacement).

sr Pages scanned by page-replacement algorithm.

cy Clock cycles by page-replacement algorithm.

Faults Trap and interrupt rate averages per second over the sampling interval.

in Device interrupts.

sy System calls.

cs Kernel thread context switches.

CPU Breakdown of percentage usage of CPU time.

us User time.

sy System time.

id CPU idle time.

wa CPU cycles to determine that the current process is waiting and there is pending disk input/output.

Disk xfer Provides the number of transfers per second to the specified physical volumes that occurred in the sample interval. The PhysicalVolume parameter can be used to specify one to four names. Transfer statistics are given for each specified drive in the order specified. This count represents requests to the physical device. It does not imply an amount of data that was read or written. Several logical requests can be combined into one physical request.

The **vmstat** command, used with the -s flag, writes to the standard output the contents of the sum structure, which contain an absolute count of paging events since system initialization. The -s option is exclusive of the other **vmstat** command options.

The following is an example of the **vmstat** command using the -s flag:

```
# vmstat -s
    8765020 total address trans. faults
    4832918 page ins
```

```
  2989263 page outs
       19 paging space page ins
        7 paging space page outs
        0 total reclaims
  5417148 zero filled pages faults
    12633 executable filled pages faults
 15031850 pages examined by clock
      118 revolutions of the clock hand
  6086090 pages freed by the clock
   105808 backtracks
        0 lock misses
        0 free frame waits
        0 extend XPT waits
  2025516 pending I/O waits
  3031667 start I/Os
  3031667 iodones
 24786000 cpu context switches
 77240518 device interrupts
        0 software interrupts
        0 traps
191650677 syscalls
```

A list of all the possible events is provided in the following.

address trans. faults Incremented for each occurrence of an address translation page fault. I/O may or may not be required to resolve the page fault. Storage protection page faults (lock misses) are not included in this count.

page ins Incremented for each page read in by the virtual memory manager. The count is incremented for page ins from page space and file space. Along with the page-out statistic, this represents the total amount of real I/O initiated by the virtual memory manager.

page outs Incremented for each page written out by the virtual memory manager. The count is incremented for page-outs to page space and for page-outs to file space. Along with the page-in statistic, this represents the total amount of real I/O initiated by the virtual memory manager.

paging space page ins Incremented for VMM-initiated page-ins from paging space only.

paging space page outs Incremented for VMM-initiated page-outs to paging space only.

total reclaims	Incremented when an address translation fault can be satisfied without initiating a new I/O request. This can occur if the page has been previously requested by VMM, but the I/O has not yet completed; or if the page was pre-fetched by VMM's read-ahead algorithm, but was hidden from the faulting segment; or if the page has been put on the free list and has not yet been reused.
zero filled pages faults	Incremented if the page fault is to working storage and can be satisfied by assigning a frame and zero-filling it.
executable filled pages faults	Incremented for each instruction page fault.
pages examined by clock	VMM uses a clock-algorithm to implement a pseudo least recently used (LRU) page replacement scheme. Pages are aged by being examined by the clock. This count is incremented for each page examined by the clock.
revolutions of the clock hand	Incremented for each VMM clock revolution (that is, after each complete scan of memory).
pages freed by the clock	Incremented for each page the clock algorithm selects to free from real memory.
backtracks	Incremented for each page fault that occurs while resolving a previous page fault. (The new page fault must be resolved first and then initial page faults can be backtracked.)
lock misses	VMM enforces locks for concurrency by removing address ability to a page. A page fault can occur due to a lock miss, and this count is incremented for each such occurrence.
free frame waits	Incremented each time a process is waited by VMM while free frames are gathered.
extend XPT waits	Incremented each time a process is waited by VMM due to a commit in progress for the segment being accessed.
pending I/O waits	Incremented each time a process is waited by VMM for a page-in I/O to complete.

start I/Os	Incremented for each read or write I/O request initiated by VMM.
iodones	Incremented at the completion of each VMM I/O request.
CPU context switches	Incremented for each CPU context switch (dispatch of a new process).
device interrupts	Incremented for each hardware interrupt.
software interrupts	Incremented for each software interrupt. A software interrupt is a machine instruction similar to a hardware interrupt that saves some state and branches to a service routine. System calls are implemented with software interrupt instructions that branch to the system call handler routine.
traps	Unused by the AIX operating system.
syscalls	Incremented for each system call.

In the following examples, an idle system will be shown, and then load will be put on to the system; the resultant output will be analyzed to investigate potential problems.

The following is the output of the **vmstat** command without any load:

```
# vmstat 1 5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 16057  1291   0   0   0   8   22   0 113  281  36  1  0 98  1
 0  0 16057  1290   0   0   0   0    0   0 108  472  25  0  0 99  0
 0  0 16057  1290   0   0   0   0    0   0 109  282  32  0  0 99  0
 0  0 16057  1290   0   0   0   0    0   0 109  285  26  0  0 99  0
 0  0 16057  1290   0   0   0   0    0   0 108  282  29  0  0 99  0
```

The first output line gives the average since system boot and can be left out when calculating system load. This is the same as running the **vmstat** command without any flags.

For the purpose of this exercise, the output of the **vmtune** command is as follows:

```
# /usr/samples/kernel/vmtune
vmtune:  current values:
  -p       -P        -r          -R         -f       -F       -N        -W
minperm  maxperm  minpgahead maxpgahead  minfree  maxfree  pd_npages maxrandwrt
  26007   104028       2          8        120      128     524288        0
```

```
  -M      -w      -k      -c        -b         -B           -u        -l    -d
maxpin npswarn npskill numclust numfsbufs hd_pbuf_cnt lvm_bufcnt lrubucket
defps
104851    4096    1024       1      93         80           9      131072     1

        -s              -n         -S           -h
sync_release_ilock  nokillroot  v_pinshm  strict_maxperm
        0               0           0             0

number of valid memory pages = 131063   maxperm=79.4% of real memory
maximum pinable=80.0% of real memory    minperm=19.8% of real memory
number of file memory pages = 101629    numperm=77.5% of real memory
```

The **vmstat** command, with only an interval parameter and count parameter, is as follows:

```
# vmstat 1 15
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 16299  1749   0   0   0   8   21   0 113  281  36  1  0 98  1
 1  1 16299  1529   0   0   0   0    0   0 301 8707 382 52 13  0 35
 1  1 16299  1398   0   0   0   0    0   0 185 6557 238 91  8  0  1
 1  1 16299  1227   0   0   0   0    0   0 225 6705 257 85 15  0  0
 1  0 16299  1049   0   0   0   0    0   0 246 6587 334 71 10  0 19
 1  1 16299   861   0   0   0   0    0   0 250 9051 317 72 19  0  9
 0  1 16265   653   0   0   0   0    0   0 342 10304 516 37 21  0 43
 4  0 16284   121   0   0   0  16   35   0 253 2432 375 36  6 43 15
 0  0 16284   120   0   0   0 432 1066   0 265  302 246 31  4 54 11
 1  0 16284   121   0   0   0 160  389   0 221 1184 239  8  5 77 10
 0  1 16284   120   0   0   0 576 1447   0 394 2377 525 28  9 39 24
 0  0 16284   122   0   0   0 232  480   0 277 1185 346 21  5 63 11
 0  0 16284   122   0   0   0 384 1630   0 326 1081 400 16 12 51 21
 0  0 16284   126   0   0   0 336  784   0 284  742 326 20  3 59 18
 0  1 16284   126   0   0   0 761 1615   0 336 1032 420 36  4 48 12
```

As listed, kthr (kernel thread), r (runnable threads), and b (waiting threads) outputs stayed relatively constant and low. The r thread should be less than 5 under a stable workload. This b value should usually be near 0.

In the memory column, the avm (average paging space memory) stayed relatively stable but the fre (free memory frames) value dropped from 1749 to its lowest of 120. If the fre value had dropped below 120 for an extended period of time, this system would be continuously paging in and out, which would lead to system performance problems.

For the page heading, the re, pi, po, and cy values remained relatively constant. The fr and sr rates, however, increased substantially. The pi rate should not go above five; however if a page-in occurs, then there must have been a previous

page-out for that page. It is also likely in a memory-constrained environment that each page-in will force a different page to be stolen and, therefore, paged out. If the system is reading in a significant number of persistent pages, you may see an increase in po without corresponding increases in pi. This situation does not necessarily indicate thrashing, but may warrant investigation into data access patterns of the applications. The fr column represents the number of pages freed and the sr column represents the number of pages scanned by the page placement algorithm. With stable, unfragmented memory, the scan rate and free rate may be nearly equal. On systems with multiple processes using many different pages, the pages are more volatile and disjointed. In this scenario, the scan rate may greatly exceed the free rate.

For the faults heading, the in, sy, and cs values fluctuated at various intervals. There is no steadfast limit to these, as the overhead is minimal, and it is difficult to say what is excessive. The only thing to remember is that the in value will always be higher than 100.

For the cpu heading, the us, sy, id, and wa values also fluctuated dramatically. The output is in percent of CPU utilization. The us output is the amount of time spent by a process executing functions without having to use the system (kernel) mode. The sy time details the amount of time a process spends utilizing system (kernel) resources. Optimum use would have the CPU working 100 percent of the time. This holds true in the case of a single-user system with no need to share the CPU. Generally, if us + sy time is below 90 percent, a single-user system is not considered CPU constrained. However, if us + sy time on a multi-user system exceeds 80 percent, the processes may spend time waiting in the run queue. Response time and throughput might suffer. The id output is the CPU idle time. The wa output is idle time with pending local disk I/O. A wa value over 40 percent could indicate that the disk subsystem may not be balanced properly, or it may be the result of a disk-intensive workload. The four values added together will give a CPU utilization of 100 percent.

3.3 The ps command

In this section, the following topics are covered:

- Use of the **ps** command in CPU usage study
- Use of the **ps** command in memory usage study

The **ps** command determines which processes are running and the resources they use.

3.3.1 Use of the ps command in a CPU usage study

Three of the possible **ps** command output columns report CPU usage, each in a different way, as shown in Table 3-2. These columns will be the topic of discussion in the sections that follow.

Table 3-2 CPU-related ps output

Column	Value
C	Recent used CPU time for process.
TIME	Total CPU time used by the process since it started.
%CPU	Total CPU time used by the process since it started, divided by the elapsed time since the process started. This is a measure of the dependence of the program on CPU.

The C column

The C column can be generated by the -l flag and the -f flag. In this column, the CPU utilization of processes or threads is reported. The value is incremented each time the system clock ticks and the process or thread is found to be running. Therefore, it also can be said to be a process penalty for recent CPU usage. The value is decayed by the scheduler by dividing it by 2 once per second when the process is not using CPU. Large values indicate a CPU-intensive process and result in lower process priority, while small values indicate an I/O intensive process and result in a more favorable priority. In the following example, the tctestprog program is a CPU-intensive program.

Another aspect of the **ps** command is the formatted output. The following formatting sorts the output according to the third column with the largest value at the top, and shows only five lines of the total output:

```
# ps -ef | sort +3 -r |head -n 5
    UID   PID  PPID   C    STIME    TTY  TIME CMD
    root 22656 27028 101 15:18:31 pts/11  7:43 ./tctestprog
    root 14718 24618   5 15:26:15 pts/17  0:00 ps -ef
    root  4170     1   3   Jun 15      - 12:00 /usr/sbin/syncd 60
    root 21442 24618   2 15:26:15 pts/17  0:00 sort +3 -r
```

From the previous example, you can determine that tctestprog is the process using the most CPU, due to the C value of 101.

The following **vmstat** output shows that the CPU is used about 25 percent by usr processes:

```
# vmstat 2 3
kthr     memory             page              faults        cpu
```

```
----- ----------- ---------------------- ------------ -----------
 r  b  avm     fre     re  pi  po  fr  sr  cy  in   sy      cs   us  sy  id  wa
 0  0  26468   51691   0   0   0   0   0   0   100  91      6    47  0   53  0
 1  1  26468   51691   0   0   0   0   0   0   415  35918   237  26  2   71  0
 1  1  26468   51691   0   0   0   0   0   0   405  70      26   25  0   75  0
```

The TIME column

The **ps** command output column TIME is generated with all flags, and it shows the total execution time for the process. This calculation does not take into account when the process was started, as seen in the following output. The same test program is used again, and even though the C column shows that the process gets a lot of CPU time, it is not the process with the largest value in the TIME column:

```
# ps -ef | sort +3 -r |head -n 5
    UID   PID  PPID   C    STIME    TTY  TIME CMD
    root 18802 27028 120 15:40:28 pts/11  1:10 ./tctestprog
    root  9298 24618   3 15:41:38 pts/17  0:00 ps -ef
    root 15782 24618   2 15:41:38 pts/17  0:00 head -n 5
    root 24618 26172   2   Jun 21 pts/17  0:03 ksh

# ps -e |head -n 1 ; ps -e|egrep -v "TIME|0:"|sort +2b -3 -n -r|head -n 10
   PID    TTY  TIME CMD
  4170      - 12:01 syncd
  4460      -  2:07 X
  3398      -  1:48 dtsession
 18802 pts/11  1:14 tctestprog
```

The syncd, X, and dtsession are all processes that have been active since IPL; that is why they have accumulated more total TIME than the test program.

The %CPU column

The %CPU column of the **ps** command output, generated by the -u or the -v flags, shows the percentage of time the process has used the CPU since the process started. The value is computed by dividing the time the process uses the CPU by the elapsed time of the process. In a multi-processor environment, the value is further divided by the number of available CPUs, because several threads in the same process can run on different CPUs at the same time. Because the time base over which this data is computed varies, the sum of all %CPU fields can exceed 100 percent. In the example below, there are two ways to sort the extracted output from a system. The first example includes kprocs, for example, PID 516, which is a wait process. The other, more complex command syntax, excludes such kprocs:

```
# ps auxwww |head -n 5
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
root     18802 25.0  1.0 4140 4160  pts/11 A   15:40:28  5:44 ./tctestprog
```

```
root       516 25.0  5.0    8 15136      - A      Jun 15 17246:34 kproc
root       774 20.6  5.0    8 15136      - A      Jun 15 14210:30 kproc
root      1290  5.9  5.0    8 15136      - A      Jun 15 4077:38  kproc

# ps gu|head -n1; ps gu|egrep -v "CPU|kproc"|sort +2b -3 -n -r |head -n 5
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
root     18802 25.0  1.0 4140 4160 pts/11  A    15:40:28  7:11 ./tctestprog
imnadm   12900  0.0  0.0  264  332       - A   Jun 15 0:00 /usr/IMNSearch/ht
root         0  0.0  5.0   12 15140      - A      Jun 15  4:11 swapper
root         1  0.0  0.0  692  764       - A      Jun 15  0:28 /etc/init
root      3398  0.0  1.0 1692 2032       - A Jun 15  1:48 /usr/dt/bin/dtses
```

From the output, you can determine that the test program, tctestprog, has used about 25 percent of the available CPU resources since the process started.

The commonly used flags of the **ps** command are provided in Table 3-3.

Table 3-3 Commonly used flags of the ps command

Flag	Description
-A	Writes to standard output information about all processes.
-a	Writes to standard output information about all processes, except the session leaders and processes not associated with a terminal.
-c *Clist*	Displays only information about processes assigned to the workload management classes listed in the Clist variable. The Clist variable is either a comma-separated list of class names or a list of class names enclosed in double quotation marks (" "), which is separated from one another by a comma or by one or more spaces, or both.
-d	Writes information to standard output about all processes except the session leaders.
-e	Writes information to standard output about all processes except kernel processes.
-F *format*	Same as -o *format.*
-f	Generates a full listing.
-k	Lists kernel processes.
-l	Generates a long listing. See also the -l flag.
-m	Lists kernel threads as well as processes. Output lines for processes are followed by an additional output line for each kernel thread.
-o *format*	Displays information in the format specified by the format variable.
a	Displays information about all processes with terminals.

e	Displays the environment as well as the parameters to the command up to a limit of 80 characters.
g	Displays all processes.

3.3.2 Use of the ps command in a memory usage study

The **ps** command can also provide useful information on memory usage. The most useful output is presented in Table 3-4 and discussed in the following sections.

Table 3-4 Memory-related ps output

Column	Value
SIZE	The virtual size of the data section of the process in 1 KB units
RSS	The real-memory size of the process in 1 KB units
%MEM	The percentage of real memory used by this process

The SIZE column

The v flag generates the SIZE column. This is the virtual size (in the paging space), in kilobytes, of the data section of the process (displayed as SZ by other flags). This value is equal to the number of working segment pages of the process that have been touched times four. If several working segment pages are currently paged out, this number is larger than the amount of real memory being used. SIZE includes pages in the private segment and the shared-library data segment of the process, as in the following example:

```
# ps av |sort +5 -r |head -n 5
   PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
 25298 pts/10 A     0:00    0  2924    12 32768   159     0  0.0  0.0 smitty
 13160   lft0 A     0:00   17   368    72 32768    40 60  0.0 0.0/usr/sbin
 27028 pts/11 A     0:00   90   292   416 32768   198   232  0.0  1.0 ksh
 24618 pts/17 A     0:04  318   292   408 32768   198   232  0.0  1.0 ksh
```

The RSS column

The v flag also produces the RSS column, as shown in the previous example. This is the real-memory (resident set) size in kilobytes of the process. This number is equal to the sum of the number of working segment and code segment pages in memory times four. Remember that code segment pages are shared among all of the currently running instances of the program. If 26 ksh processes are running, only one copy of any given page of the ksh executable program would be in memory, but the **ps** command would report that code segment size as part of the RSS of each instance of the ksh program.

If you want to sort on the sixth column, you receive the output sorted by the RSS column, as shown in the following example:

```
# ps av |sort +6 -r |head -n 5
PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
21720  pts/1 A      0:00    1   288   568 32768   198   232  0.0  1.0 ksh
27028 pts/11 A      0:00   90   292   416 32768   198   232  0.0  1.0 ksh
24618 pts/17 A      0:04  318   292   408 32768   198   232  0.0  1.0 ksh
15698  pts/1 A      0:00    0   196   292 32768    52    60  0.0  0.0 ps av
```

The %MEM column

The %MEM column is generated by the u and the v flags. This is calculated as the sum of the number of working segment and code segment pages in memory times four (that is, the RSS value), divided by the size of the real memory of the machine in KB, times 100, rounded to the nearest full percentage point. This value attempts to convey the percentage of real memory being used by the process. Unfortunately, it tends to exaggerate the cost of a process that is sharing program text with other processes, such as with RSS. The rounding to the nearest percentage point causes all of the processes in the system that have RSS values under .005 times real memory size to have a %MEM of 0.0.

The following is an example of the **ps** %MEM column:

```
# ps au |head -n 1; ps au |egrep -v "RSS"|sort +3 -r |head -n 5
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
root    22750  0.0 21.0 20752 20812 pts/11 A 17:55:51  0:00./tctestprog2 root
21720  0.0  1.0  484  568  pts/1 A    17:16:14  0:00 ksh
root     25298  0.0  0.0 3080   12 pts/10 A      Jun 16  0:00 smitty
root     27028  0.0  0.0  488  416 pts/11 A    14:53:27  0:00 ksh
root     24618  0.0  0.0  488  408 pts/17 A      Jun 21  0:04 ksh
```

You can combine all of these columns into one output by using the gv flags. For example:

```
# ps gv|head -n 1; ps gv|egrep -v "RSS" | sort +6b -7 -n -r |head -n 5
PID    TTY STAT  TIME PGIN SIZE RSS   LIM  TSIZ TRS %CPU %MEM COMMAND
15674 pts/11 A   0:01    0   36108 36172  32768   5    24  0.6 24.0 ./tctestp
22742 pts/11 A   0:00    0   20748 20812  32768  5    24  0.0 14.0 ./backups
10256  pts/1 A   0:00    0   15628 15692  32768  5    24  0.0 11.0 ./tctestp
2064       -  A   2:13    5      64  6448     xx   0  6392  0.0  4.0 kproc
1806       -  A   0:20    0      16  6408     xx   0  6392  0.0  4.0 kproc
```

In the following list, additional columns in the previous examples are explained:

PGIN The number of page-ins caused by page faults. Since all I/O operations are classified as page faults by the **ps** command, this is a measure of I/O volume.

TSIZ The size of the text (shared-program) image. This is the size of the text section of the executable file. Pages of the text section of the executable program are only brought into memory when they are touched, that is, branched to or loaded from. This number represents only an upper bound on the amount of text that could be loaded. The TSIZ value does not reflect actual memory usage. This TSIZ value can also be seen by executing the dump -ov command against an executable program (for example, dump -ov /usr/bin/ls).

TRS The size of the resident set (real memory) of text. This is the number of code segment pages multiplied by four. This number exaggerates the memory used for programs where multiple instances are running. The TRS value can be higher than the TSIZ value because other pages may be included in the code segment, such as the XCOFF header and the loader section.

3.4 The tprof command

In this section the following topics are discussed:

- The use of **tprof** to study general CPU performance
- The use of **tprof** on a user program

3.4.1 Using the tprof general report

In the AIX operating system, an interrupt occurs periodically to allow a *housekeeping* kernel routine to run. This occurs 100 times per second. When the **tprof** command is invoked, it counts every such kernel interrupt as a *tick*. This kernel routine records the process ID and the address of the instruction executing when the interrupt occurs, for use by the **tprof** command. The **tprof** command also records whether the process counter is in the kernel address space, user address space, or shared library address space.

A summary ASCII report with the suffix .all is always produced. If no program is specified, the report is named __prof.all. If a program is specified, the report is named __*program*.all. This report contains an estimate of the amount of CPU time spent in each process that was executing while the **tprof** program was monitoring the system. It also contains an estimate of the amount of CPU time spent in each of the three address spaces and the amount of time the CPU was idle.

The **tprof** command uses the AIX Kernel trace facility. Only one user can use the AIX trace facility at a time. Thus, only one **tprof** command can be active in the system at a time.

The files containing the reports are left in the working directory. All files created by the **tprof** command are prefixed by ___ (two underscores).

In the following example, a generic report is generated:

```
# tprof -x sleep 30
Starting Trace now
Starting  sleep 30
Wed Jun 28 14:58:58 2000
System: AIX server3 Node: 4 Machine: 000BC6DD4C00

Trace is done now
30.907 secs in measured interval
 * Samples from __trc_rpt2
 * Reached second section of __trc_rpt2
```

In this case, the sleep 30 parameter directs the **tprof** command to run for 30 seconds.

The Total column in the __prof.all is useful. The first section indicates the use of ticks on a per process basis.

```
Process      PID      TID     Total   Kernel User   Shared    Other
=======      ===      ===     =====   ====== ====   ======    =====
wait          516      517      3237     3237   0        0        0
tctestprg   14746    13783      3207        1   3206     0        0
tctestprg   13730    17293      3195        0   3195     0        0
wait         1032     1033      3105     3105   0        0
wait         1290     1291       138      138   0        0        0
swapper         0        3        10        7   3        0        0
tprof       14156     5443         6        3   3        0        0
trace       16000    14269         3        3   0        0        0
syncd        3158     4735         2        2   0        0        0
tprof        5236    16061         2        2   0        0        0
gil          2064     2839         1        1   0        0        0
gil          2064     3097         1        1   0        0
trace       15536    14847         1        1   0        0        0
sh          14002    16905         1        1   0        0        0
sleep       14002    16905         1        1   0        0        0
=======      ===      ===     =====   ======     ====   ======    =====
Total                        12910      6503     6407        0        0
```

Since each tick is 1/100 second, 30 seconds requires a total of 3000 ticks. However, when looking at the output, there are over 12000 total ticks. The result

is dependent on the hardware, in this case a four-way F50, so the available ticks are calculated in the following way:

```
Time (in seconds) x number of available CPUs x 100
```

In the previous output, you can determine that both **tctestprg** processes use about 3200 ticks, approximately 25 percent of the total available ticks. This is confirmed with the following **ps auxwww** output:

```
# ps auxwww
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
root     14020 25.0  0.0  300  320  pts/1 A    15:23:55 16:45 ./tctestprg
root     12280 25.0  0.0  300  320  pts/1 A    15:23:57 16:43 ./tctestprg
```

In the second section of the general report of the **tproc** command, the total amount of ticks used by a specified type of process is noted. In this section, the total ticks used by a process type are in the Total column, and the number of processes representing that type can be seen in the FREQ column.

```
Process      FREQ    Total   Kernel     User   Shared    Other
=======      ===    =====   ======     ====   ======    =====
wait           3     6480     6480        0        0        0
tctestprg      2     6402        1     6401        0        0
swapper        1       10        7        3        0        0
tprof          2        8        5        3        0        0
trace          2        4        4        0        0        0
gil            2        2        2        0        0        0
syncd          1        2        2        0        0        0
sh             1        1        1        0        0        0
sleep          1        1        1        0        0        0
=======      ===    =====   ======     ====   ======    =====
Total         15    12910     6503     6407        0        0
```

3.4.2 Using tprof on a program

The **tprof** command is also a useful tool for C, C++, or FORTRAN programs that might be CPU bound. It identifies sections of a program that are most heavily using the CPU. The **tprof** command executes a program and then produces a set of files containing reports. The reports are divided down to subroutine level.

The following example is a basic one; there are many more possibilities outside the scope of this document:

```
# tprof ./tctestprg
Starting Trace now
Starting  ./tctestprg
Wed Jun 28 15:57:35 2000
System: AIX server3 Node: 4 Machine: 000BC6DD4C00
Trace is done now
```

```
23.258 secs in measured interval
 * Samples from __trc_rpt2
 * Reached second section of __trc_rpt2
(The tctestprg process was manually killed)
```

The output file is named __tctestprg.all, and the output is restricted to the process provided as an argument to the command. The first section is the same as shown in a general report. However, it only reflects a single process.

```
# more __tctestprg.all
Process      PID      TID    Total   Kernel     User   Shared    Other
=======      ===      ===    =====   ======     ====   ======    =====
./tctestprg 16276    16081     2156        0     2156        0        0
=======      ===      ===    =====   ======     ====   ======    =====
Total                           2156        0     2156        0        0
```

The second section is a trivial report on the single process.

```
Process     FREQ    Total   Kernel     User   Shared    Other
=======      ===    =====   ======     ====   ======    =====
./tctestprg  1     2156        0     2156        0        0
=======      ===    =====   ======     ====   ======    =====
Total        1     2156        0     2156        0        0
```

The third section provides information about the subroutines used in the specified process providing information about areas that require tuning.

```
Total Ticks For ./tctestprg (USER) = 2156
Subroutine               Ticks  %   Source   Address Bytes
=============            ===== ==== ======== ======== ======
.main                     1368 14.5 case.c   10000318 4c
.casework                  788  8.4 case.c   10000364 54
```

3.5 The svmon command

The **svmon** command is used to display information regarding the current memory state. Although it is a complicated command, you should understand what it can do to assist you in performance monitoring.

The **svmon** command generates seven types of reports:

- Global
- User
- Process
- Segment
- Detailed segment

- Command
- Workload management class

To run each of these reports, a report indicator flag needs to be used.

With the exception of the **svmon -G** and **svmon -D** reports, the other report options use the same flags with a similar use. In the following sections, an example of the command and the output generated is provided. This is not an exhaustive list of functions; rather, it is a short demonstration of the versatility of the **svmon** command. For an explanation of the flags, see 3.5.8, “The svmon command flags” on page 90.

3.5.1 The svmon global report

The global report is generated when the -G flag is specified.

The **svmon -G** command has the following syntax:

```
svmon -G [ -i interval [ NumIntervals] ] [ -z ]
```

Running the **svmon -G** global report command generates the following output:

```
# svmon -G

               size      inuse       free        pin    virtual
memory       131063     119922      11141       6807      15924
pg space     131072        305

               work       pers       clnt
pin             6816          0          0
in use         21791      98131          0
```

The **svmon -G** command with an interval and the number of intervals, and providing the -z flag to obtain the maximum memory allocated, provides the following output:

```
# svmon -G -i1 5 -z

               size      inuse       free        pin    virtual
memory       131063     125037       6026       6811      15948
pg space     131072        305

               work       pers       clnt
pin             6820          0          0
in use         21950     103087          0


               size      inuse       free        pin    virtual
memory       131063     125847       5216       6811      15949
```

```
pg space     131072        305

              work       pers       clnt
pin           6820          0          0
in use       21954     103893          0


              size      inuse       free        pin    virtual
memory      131063     126769       4294       6811      15949
pg space    131072        305

              work       pers       clnt
pin           6820          0          0
in use       21954     104815          0


              size      inuse       free        pin    virtual
memory      131063     127890       3173       6811      15949
pg space    131072        305

              work       pers       clnt
pin           6820          0          0
in use       21954     105936          0


              size      inuse       free        pin    virtual
memory      131063     129092       1971       6811      15949
pg space    131072        305

              work       pers       clnt
pin           6820          0          0
in use       21954     107138          0

Maximum memory allocated = 432
```

In the previous example, the headings have the following meanings:

memory Specifies statistics describing the use of real memory, including:

- **size** Number of real memory frames (size of real memory)
- **inuse** Number of frames containing pages
- **free** Number of frames free
- **pin** Number of frames containing pinned pages
- **virtual** Number of pages allocated in the system virtual space

stolen	Number of frames stolen by **rmss** (Reduced-Memory System Simulator; see 3.6, “The rmss command” on page 92, for more information) and made unusable by the VMM
pg space	Specifies statistics describing the use of paging space. This data is reported only if the -r flag is not used.
size	Size of paging space
inuse	Number of paging space pages in use
pin	Specifies statistics on the subset of real memory containing pinned pages, including:
work	Number of frames containing pinned pages from working segments
pers	Number of frames containing pinned pages from persistent segments
clnt	Number of frames containing pinned pages from client segments
in use	Specifies statistics on the subset of real memory in use, including:
work	Number of frames containing pages from working segments
pers	Number of frames containing pages from persistent segments
clnt	Number of frames containing pages from client segments

3.5.2 The svmon user report

The user report is printed when the -U flag is specified.

The **svmon -U** command has the following syntax:

```
svmon -U [ lognm1...lognmN] [ -n | -s ] [ -w | -f | -c ] [ -t count ]
[ -u | -p | -g | -v ] [ -i interval [ NumIntervals]] [ -l ] [ -d ] [ -z ]
[ -m ]
```

The **svmon -U** without any options produces output similar to the following:

```
# svmon -U
===============================================================================
User root               Inuse      Pin     Pgsp  Virtual
                        18447     1327      175     7899
```

```
.....................................................................
SYSTEM segments          Inuse      Pin     Pgsp  Virtual
                          3816     1269      175     3535

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3792  1265  175  3511   0..32767 :

65475..65535
  9352        - work                          12     1    0    12   0..49746
   220        - work                           6     1    0     6   0..49746
  7a0f        - work                           4     1    0     4   0..49746
  502a        - work                           2     1    0     2   0..49746

.....................................................................
EXCLUSIVE segments       Inuse      Pin     Pgsp  Virtual
                        12551       58        0     3891

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  7162        - pers /dev/lv00:17           6625     0    -     -   0..100987
...
2b65        - pers /dev/hd4:4294             0     0    -     -
  1369        - pers /dev/hd3:32               0     0    -     -
  1b63        1 pers code,/dev/hd2:4536        0     0    -     -   0..21
  5b4b        1 pers code,/dev/hd2:10545       0     0    -     -   0..1
  1b43        - pers /dev/hd2:32545            0     0    -     -   0..0
  e6ff        - pers /dev/hd4:732              0     0    -     -
  3326        1 pers code,/dev/hd2:10553       0     0    -     -   0..4
  2ee6        - pers /dev/hd2:14469            0     0    -     -
  ea9d        - pers /dev/hd2:39225            0     0    -     -   0..4
  d67b        - pers /dev/hd2:32715            0     0    -     -   0..0
  5668        - pers /dev/hd2:98694            0     0    -     -   0..0
  466a        1 pers code,/dev/hd2:98696       0     0    -     -   0..4
  d21a        1 pers code,/dev/hd2:10679       0     0    -     -   0..1
   a41        - pers /dev/hd2:32224            0     0    -     -   0..1
  aa15        1 pers code,/dev/hd2:10673       0     0    -     -   0..0
  f1fe        - pers /dev/hd2:10310            0     0    -     -   0..2
  e9fd        - pers /dev/hd2:10309            0     0    -     -   0..14
  c9f9        - pers /dev/hd2:32705            0     0    -     -   0..3
  b9f7        1 pers code,/dev/hd2:10734       0     0    -     -   0..15
  a1f4        1 pers code,/dev/hd2:10765       0     0    -     -   0..10
  3a07        1 pers code,/dev/hd2:10684       0     0    -     -   0..7
  2a05        1 pers code,/dev/hd2:10718       0     0    -     -   0..170
  59eb        - pers /dev/hd2:32701            0     0    -     -   0..9
  e9bd        1 pers code,/dev/hd2:4123        0     0    -     -   0..128
...
=====================================================================
User guest               Inuse      Pin     Pgsp  Virtual
                             0        0        0        0
```

```
================================================================================
User nobody             Inuse      Pin     Pgsp  Virtual
                            0        0        0        0

================================================================================
User lpd                Inuse      Pin     Pgsp  Virtual
                            0        0        0        0

================================================================================
User nuucp              Inuse      Pin     Pgsp  Virtual
                            0        0        0        0

================================================================================
User ipsec              Inuse      Pin     Pgsp  Virtual
                            0        0        0        0

================================================================================
User netinst            Inuse      Pin     Pgsp  Virtual
                            0        0        0        0
```

To check a particular users' utilization, as well as the total memory allocated, use the following command:

```
# svmon -U root -z

================================================================================
User root               Inuse      Pin     Pgsp  Virtual
                        10980     1322      175     7913

--------------------------------------------------------------------------------
SYSTEM segments         Inuse      Pin     Pgsp  Virtual
                         3816     1269      175     3535

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3792  1265  175  3511   0..32767 :

65475..65535
  9352        - work                          12     1    0    12   0..49746
   220        - work                           6     1    0     6   0..49746
  7a0f        - work                           4     1    0     4   0..49746
  502a        - work                           2     1    0     2   0..49746

--------------------------------------------------------------------------------
EXCLUSIVE segments      Inuse      Pin     Pgsp  Virtual
                         5024       53        0     3905

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  1be3        2 work process private         580     8    0   579   0..675 :
...
```

```
d9fb        - pers /dev/hd9var:86           0     0    -     -   0..0
  c9f9        - pers /dev/hd2:32705           0     0    -     -   0..3
  a1f4        1 pers code,/dev/hd2:10765      0     0    -     -   0..10
  3a07        1 pers code,/dev/hd2:10684      0     0    -     -   0..7
  2a05        1 pers code,/dev/hd2:10718      0     0    -     -   0..170
  d9bb        1 pers code,/dev/hd2:4379       0     0    -     -   0..20
  c955        - pers /dev/hd3:33              0     0    -     -   0..5
  4168        - pers /dev/hd2:20485           0     0    -     -   0..0
  2965        - pers /dev/hd2:20486           0     0    -     -   0..7
  694d        - pers /dev/hd9var:2079         0     0    -     -   0..0
  514a        - pers /dev/hd9var:2078         0     0    -     -   0..0
  30a6        - pers /dev/hd9var:2048         0     0    -     -   0..0
  4088        - pers /dev/hd2:4098            0     0    -     -   0..1
  dbfb        - pers /dev/hd3:21              0     0    -     -

................................................................................
SHARED segments         Inuse      Pin     Pgsp  Virtual
                         2140        0        0      473

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  8811        d work shared library text    2080     0    0   473   0..65535
  e03c        1 pers code,/dev/hd2:4204       58     0    -     -   0..58
  2865        - pers /dev/hd2:32343            2     0    -     -   0..1
Maximum memory allocated = 21473
```

The headings have the following meaning:

User Indicates the user name

Inuse Indicates the total number of pages in real memory in segments that are used by the user

Pin Indicates the total number of pages pinned in segments that are used by the user

Pgsp Indicates the total number of pages reserved or used on paging space by segments that are used by the user

Virtual Indicates the total number of pages allocated in the process virtual space

Once the column heading is displayed, **svmon** displays (if the -d flag is specified) information about all the processes run by the specified login user name. It only contains the column heading of the processes, as described in the process report.

Then **svmon** displays information about the segments used by those processes.

This set of segments is separated into three categories:

1. The segments that are flagged as system segments that are shared by all processes
2. The segments that are only used by the set of processes
3. The segments that are shared between several users

If the -l flag is specified, then for each segment in the last category, the list of process identifiers that use the segment is displayed. The login user name that executes the process identifier is also displayed.

3.5.3 The svmon process report

The process report is generated when the -P flag is specified.

The **svmon -P** command has the following syntax:

```
svmon [-P [pid1...pidn] [ -u | -p | -g | -v ] [ -n | -s ] [ -w | -f | -c]
[ -t count ] [ -i interval [ NumIntervals ] ] [ -l ] [ -z ] [ -m ] ]
```

The **svmon -P** command process report has output similar to the following:

```
# svmon -P | pg

------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   11126 backbyname     32698     1266      175     4114        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  7162        - pers /dev/lv00:17          26650     0    -     -   0..100362
     0        0 work kernel seg             3790  1265  175  3509   0..32767 :

65475..65535
  8811        d work shared library text    2030     0    0   540   0..65535
  c373        - pers /dev/hd3:2061           134     0    -     -   0..133
  4823        2 work process private          48     1    0    48   0..47 :

65310..65535
  2969        f work shared library data      22     0    0    17   0..749
  cdb7        3 pers shmat/mmap,/dev/hd2:     16     0    -     -   0..16
  6d28        1 pers code,/dev/hd2:10334       7     0    -     -   0..6
  5920        - pers /dev/hd2:32166            1     0    -     -   0..0

------------------------------------------------------------------------------
...
------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
    3452 telnetd         6001     1266      175     4214        N        N
```

```
  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3790  1265  175  3509   0..32767 :

65475..65535
  8811        d work shared library text    2030     0    0   540   0..65535
  3f24        2 work process private         106     1    0   106   0..96 :

65306..65535
  fa3f        f work shared library data      73     0    0    58   0..640
  d67b        - pers /dev/hd2:32715            1     0    -     -   0..0
  3406        3 work shmat/mmap                1     0    0     1   0..0
  9c13        1 pers code,/dev/hd2:10763       0     0    -     -   0..101

-------------------------------------------------------------------------------
...
-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
    6968 rtcmd           3794     1266      175     3513        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3790  1265  175  3509   0..32767 :

65475..65535
  6a0d        2 work process private           4     1    0     4
65314..65535
-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
     516 wait            3792     1266      175     3511        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3790  1265  175  3509   0..32767 :

65475..65535
  8010        2 work process private           2     1    0     2
65339..65535
-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
       0                    3        1        0        3        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  780f        2 work process private           3     1    0     3
65338..65535
```

The **svmon -P** command can be used to determine the top 10 processes using memory, sorted in decreasing order, by the total pages reserved or being used with the following command:

```
# svmon -Pv -t 10 | pg
```

```
-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   10294 X               6579     1275      175     4642        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3792  1265  175  3511   0..32767 :

65475..65535
  1be3        2 work process private         580     8    0   579   0..675 :

65309..65535
  8811        d work shared library text    2080     0    0   473   0..65535
  f3fe        f work shared library data      54     0    0    39   0..310
  4c09        - work                         32     0    0    32   0..32783
  2be5        3 work shmat/mmap               4     2    0     4   0..32767
  472b        - work                          2     0    0     2   0..32768
  2647        - work                          2     0    0     2   0..32768
  e15c        1 pers code,/dev/hd2:18475     32     0    -     -   0..706
  4168        - pers /dev/hd2:20485           0     0    -     -   0..0
  2965        - pers /dev/hd2:20486           0     0    -     -   0..7
  694d        - pers /dev/hd9var:2079         0     0    -     -   0..0
  514a        - pers /dev/hd9var:2078         0     0    -     -   0..0
  9092        - pers /dev/hd4:2               1     0    -     -   0..0
  dbfb        - pers /dev/hd3:21              0     0    -     -
...
Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3792  1265  175  3511   0..32767 :

65475..65535
  8811        d work shared library text    2080     0    0   473   0..65535
  500a        2 work process private         122     1    0   122   0..122 :

65306..65535
    20        f work shared library data      57     0    0    43   0..425
  b156        - pers /dev/hd4:4286            1     0    -     -   0..0
  d81b        1 pers code,/dev/hd2:10393      9     0    -     -   0..8

-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
    5682 sendmail: a     6081     1266      175     4136        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3792  1265  175  3511   0..32767 :

65475..65535
  8811        d work shared library text    2080     0    0   473   0..65535
  51ea        2 work process private         107     1    0   106   0..103 :
```

```
65308..65535
  29e5        f work shared library data      60     0    0    46   0..417
  71ee        1 pers code,/dev/hd2:10755      38     0    -     -   0..106
  59eb        - pers /dev/hd2:32701            4     0    -     -   0..9
```

Each column heading has the following meaning.

Pid Indicates the process ID.

Command Indicates the command the process is running.

Inuse Indicates the total number of pages in real memory from segments that are used by the process.

Pin Indicates the total number of pages pinned from segments that are used by the process.

Pgsp Indicates the total number of pages used on paging space by segments that are used by the process. This number is reported only if the -r flag is not used.

Virtual Indicates the total number of pages allocated in the process virtual space.

64-bit Indicates if the process is a 64-bit process (Y) or a 32-bit process (N).

Mthrd Indicates if the process is multi-threaded (Y) or not (N).

3.5.4 The svmon segment report

The segment report is printed when the -S flag is specified.

The **svmon -S** command has the following syntax:

```
svmon [-S [sid1...sidn] [ -u | -p | -g | -v ] [ -n | -s ] [ -w | -f | -c ]
[ -t count ] [ -i interval [N umIntervals ] ] [ -l ] [ -z ] [ -m ] ]
```

The **svmon -S** command produces the following output:

```
# svmon -S
Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  7162        - pers /dev/lv00:17           7638     0    -     -   0..100362
  680d        - work misc kernel tables     3819     0    0  3819   0..17054 :

63488..65535
     0        - work kernel seg             3792  1265  175  3511   0..32767 :

65475..65535
  82b0        - pers /dev/hd2:26992         2390     0    -     -   0..2389
  8811        - work                        2080     0    0   473   0..65535
```

```
...
6be5        - pers /dev/hd2:153907           0     0    -     -   0..2
  67e6        - pers /dev/hd2:47135            0     0    -     -   0..1
  8fdc        - pers /dev/hd2:22746            0     0    -     -   0..0
  7be1        - pers /dev/hd2:53296            0     0    -     -   0..12
  87de        - pers /dev/hd2:69859            0     0    -     -   0..0
```

To check the memory usage of the top five working segments according to the number of virtual pages, use the following command:

```
# svmon -S -t 5 -w -v

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  680d        - work misc kernel tables     4197     0    0  4197   0..17064 :

63488..65535
     0        - work kernel seg             3797  1270  175  3516   0..32767 :

65475..65535
  700e        - work kernel pinned heap     1919   624    0  1920   0..65535
  37ad        - work                         770     1    0   770   0..764 :

65313..65535
   a8a        - work                         770     1    0   770   0..927 :

65250..65535
```

To print out the memory usage statistics of segments sorted by the number of reserved paging space blocks, use the following command:

```
# svmon -S 680d 700e -g

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  700e        - work kernel pinned heap      1919   624    0  1920   0..65535
  680d        - work misc kernel tables     4197     0    0  4197   0..17064 :

63488..65535
```

Each column heading has the following meaning.

Vsid Indicates the virtual segment ID. Identifies a unique segment in the VMM.

Esid Indicates the effective segment ID. When provided, it indicates how the segment is used by the process. If the VSID segment is mapped by several processes, but with different ESID values, then this field contains a -. In that case, the exact ESID values can be obtained through the

-P option applied on each process identifier using the segment.

Type Identifies the type of the segment: `pers` indicates a persistent segment, `work` indicates a working segment, `clnt` indicates a client segment, `map` indicates a mapped segment, and `rmap` indicates a real memory mapping segment.

Description Specifies a textual description of the segment. The value of this column depends on the segment type. If the segment is a persistent segment and is not associated with a log, then the device name and inode number of the associated file are displayed, separated by a colon. (The device name and inode can be translated into a file name with the **ncheck** command.) If the segment is the primary segment of a large file, then the words `large file` are prepended to the description. If the segment is a persistent segment and is associated with a log, then the string log is displayed.

If the segment is a working segment, then the **svmon** command attempts to determine the role of the segment. For instance, special working segments, such as the kernel and shared library, are recognized by the **svmon** command. If the segment is the private data segment for a process, then `private` is printed out. If the segment is the code segment for a process, and the segment report is printed out in response to the -P flag, then the word `code` is prepended to the description.

If the segment is mapped by several processes and used in a different way (for example, a process private segment mapped as shared memory by another process), then the description is empty. The exact description can be obtained through -P flag applied on each process identifier using the segment.

If a segment description is too large to fit in the description space then the description is truncated. The truncated part can be obtained through the -S flag (without -l) on the given segment.

Inuse Indicates the number of pages in real memory in this segment.

Pin Indicates the number of pages pinned in this segment.

Pgsp Indicates the number of pages used on paging space by this segment. This field is relevant only for working segments.

Virtual Indicates the number of pages allocated for the virtual space of the segment. (Only for working segments.)

VMM manages this value for statistical purposes. It may not be updated. Then the value may be less than the inuse counters.

Address Range Specifies the range(s) the segment pages has been allocated. The working segment may have two ranges, because pages are allocated by starting from both ends and moving towards the middle.

If the -l flag is present, the list of process identifiers that use that segment is displayed. See the -l flag description for special segments processing.

3.5.5 The svmon detailed segment report

The -D flag is used to get a more detailed listing of a segment.

The syntax of the **svmon -D** command is as follows:

```
svmon [-D sid1...sidn [-b] [ -i interval [ NumIntervals] ] [ -z ] ]
```

To print out the frames belonging to a segment, the command is as follows:

```
# svmon -D 700e

Segid: 700e
Type:  working
Address Range: 0..65535
Size of page space allocation: 0 pages (  0.0 Mb)
Virtual: 1920 frames ( 7.5 Mb)
Inuse: 1919 frames ( 7.5 Mb)

           Page      Frame    Pin
          65471        313     Y
          65535        311     N
              0        314     Y
              1        309     Y
              2        308     Y
              3        305     Y
              4        296     Y
              5        299     Y
              6        294     Y
              7        297     Y
```

```
              8        292     Y
              9        295     Y
             10        290     Y
...
381      81019     N
            380     115074     N
            379      80725     N
           3335      57367     Y
           3336      59860     Y
           3337     107421     N
           3338     114966     N
           3339     107433     N
           3341      95069     Y
           3342      70192     Y
```

To print out the frames belonging to a segment with the status bit of each frame, use the following command:

```
# svmon -D 700e -b

Segid: 700e
Type:  working
Address Range: 0..65535
Size of page space allocation: 0 pages (  0.0 Mb)
Virtual: 1920 frames ( 7.5 Mb)
Inuse: 1919 frames ( 7.5 Mb)

           Page      Frame    Pin        Ref        Mod
          65471        313     Y          Y          Y
          65535        311     N          N          Y
              0        314     Y          N          Y
              1        309     Y          N          Y
              2        308     Y          Y          Y
              3        305     Y          Y          Y
              4        296     Y          N          Y
              5        299     Y          N          Y
              6        294     Y          N          Y
              7        297     Y          N          Y
              8        292     Y          N          Y
              9        295     Y          N          Y
             10        290     Y          N          Y
...
381      81019     N          N          Y
            380     115074     N          N          Y
            379      80725     N          N          Y
           3335      57367     Y          N          Y
           3336      59860     Y          N          Y
           3337     107421     N          N          Y
           3338     114966     N          N          Y
```

```
3339      107433     N          N          Y
3341       95069     Y          N          Y
3342       70192     Y          Y          Y
```

The output headings have the following meanings. The segid, type, address range, size of page space allocation, and virtual and inuse headings are explained at the end of 3.5.4, “The svmon segment report” on page 80.

Page Relative page number to the virtual space. This page number can be higher than the number of frames in a segment (65532) if the virtual space is larger than a single segment (large file).

Frame Frame number in real memory.

Pin Indicates if the frame is pinned or not.

Ref Indicates if the frame has been referenced by a process (-b option only).

Mod Indicates if the frame has been modified by a process (-b option only).

3.5.6 The svmon command report

The command report provides a usage summary of specific commands being run. The command report is printed when the -C flag is specified.

The **svmon -C** command has the following syntax:

```
svmon [-C cmd1...cmdn [ -u | -p | -g | -v ] [ -n | -s [ -w | -f | -c ]
[ -t count] [ -i interval [ NumIntervals ] ] [ -d ] [ -l ] [ -z ] [ -m ] ]
```

To check the memory usage of specific commands, use the following command:

```
# svmon -C savevg ftp
# pg /tmp/file

===============================================================================
Command ftp             Inuse      Pin     Pgsp  Virtual
                        42104     1271      175     3966

...............................................................................
SYSTEM segments         Inuse      Pin     Pgsp  Virtual
                         3798     1270      175     3517

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3798  1270  175  3517   0..32767 :

65475..65535
...............................................................................
```

```
EXCLUSIVE segments      Inuse      Pin     Pgsp  Virtual
                        36189        1        0      148

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  985e        - pers /dev/lv00:17          35977     0    -     -   0..40307
  322a        2 work process private         112     1    0   109   0..83 :

65257..65535
   22c        f work shared library data      53     0    0    39   0..849
   64e        1 pers code,/dev/hd2:4550       44     0    -     -   0..57
1c88        - pers /dev/hd2:32628            3     0    -     -   0..2

................................................................................
SHARED segments         Inuse      Pin     Pgsp  Virtual
                         2117        0        0      301

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  8811        d work shared library text    2117     0    0   301   0..65535

================================================================================
Command savevg          Inuse      Pin     Pgsp  Virtual
savevg   *** command does not exist ***
```

If a command does not own a memory segment, the error, as shown above, will be provided.

To check a command and display the memory statistics for the command, enter the following:

```
# svmon -C ftp -d

================================================================================
Command ftp             Inuse      Pin     Pgsp  Virtual
                        46435     1266      175     3966

--------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
    2728 ftp            46435     1266      175     3966        N        N

................................................................................
SYSTEM segments         Inuse      Pin     Pgsp  Virtual
                         3798     1265      175     3517

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             3798  1265  175  3517   0..32767 :

65475..65535
................................................................................
EXCLUSIVE segments      Inuse      Pin     Pgsp  Virtual
```

```
                      40520        1        0      148

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  985e        - pers /dev/lv00:17          40308     0    -     -   0..40307
322a        2 work process private         112     1    0   109   0..83 :

65257..65535
   22c        f work shared library data      53     0    0    39   0..849
   64e        1 pers code,/dev/hd2:4550       44     0    -     -   0..57
  1c88        - pers /dev/hd2:32628            3     0    -     -   0..2

................................................................................
SHARED segments         Inuse      Pin     Pgsp  Virtual
                         2117        0        0      301

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  8811        d work shared library text    2117     0    0   301   0..65535
```

The column headings have the following meanings:

Command Indicates the command name.

Inuse Indicates the total number of pages in real memory in segments that are used by the command (all process running the command).

Pin Indicates the total number of pages pinned in segments that are used by the command (all process running the command).

Pgsp Indicates the total number of pages reserved or used on paging space by segments that are used by the command.

Virtual Indicates the total number of pages allocated in the virtual space of the command. Once this column heading is displayed, **svmon** displays (if the -d flag is specified) information about all the processes running the specified command. It only contains the column headings of the processes, as described in a process report. Then **svmon** displays information about the segments used by those processes.

This set of segments is separated into three categories: The segments that are flagged as systems that are shared by all processes the segments that are only used by the set of processes, and the segments that are shared between several command names.

If the -l flag is specified, then for each segment in the last category, the list of process identifiers that use the segment is displayed. The command name that the process identifier runs is also displayed. See the -l flag description for special segment processing.

3.5.7 The svmon Workload Manager (WLM) class report

The WLM (see 6.6, “Workload Manager (WLM)” on page 227, for information on WLM) class report is printed when the -W flag is specified. The **svmon -W** command has the following syntax:

```
svmon [ -W [class1...classn] [ -u | -p | - g | -v ] [ -n | -s ]
[ -w | -f | -c ] [ -t count ] [ -i interval [ NumIntervals ] ] [ -l ] [ -z ]
[ -m ] ]
```

Note: The **svmon -W** command should be run when Workload Manager is started in order to create the printed workload class report.

To check the Workload Manager class statistics, enter the following command:

```
# svmon -W backup
================================================================================
Superclass                Inuse              Pin  Pgsp   Virtual
backup                    52833              10   0      50329

Vsid      Esid Type    Description           Inuse  Pin    Pgsp  Virtual
6784      - work                             27989  0      0     28017
1aa18     - work                             21887  0      0     21887
14356     - pers       /dev/lv_wlm1:17       1250   0      -     -
173f5     - pers       /dev/lv_wlm2:17       1250   0      -     -
5347      - work                             103    2      0     101
c34e      - work                             77     0      0     77
1891a     - work                             77     0      0     77
14636     - work                             46     0      0     37
5327      - work                             28     0      0     20
1d83f     - work                             16     0      0     18
1e33c     - work                             16     0      0     13
10772     - work                             15     0      0     13
6a84      - work                             15     0      0     13
15457     - work                             14     0      0     14
38a1      - work                             8      0      0     8
126f0     - work                             8      0      0     8
```

```
================================================================================
Superclass              Inuse              Pin  Pgsp   Virtual
11313        - pers     /dev/hd1:26          6     0        -     -

e50c         - work                          5     2        0     5
b549         - work                          5     2        0     5
12e3         - work                          3     2        0     3
13351        - work                          3     0        0     3
14a16        - work                          3     0        0     0
12970        - work                          3     0        0     5
6904         - work                          2     2        0     2
a9c8         - work                          1     0        0     3
2320         - pers     /dev/hd1:32          1     0        -     -

1d39f        - pers     /dev/hd2:16870       1     0        -     -
834a         - pers     /dev/hd1:23          1     0        -     -
```

The column headings in a Workload Manager class report have the following meaning:

Class Indicates the workload class name.

Inuse Indicates the total number of pages in real memory in segments belonging to the workload class.

Pin Indicates the total number of pages pinned in segments belonging to the workload class.

Pgsp Indicates the total number of pages reserved or used on paging space by segments belonging to the workload class.

Virtual Indicates the total number of pages allocated in the virtual space of the workload class.

Once this column's heading is displayed, **svmon** displays information about the segments belonging to the workload class.

If the -l option is specified, then for each segment, the list of process identifiers that use the segment is displayed. The workload class the process belongs to is also displayed. See the -l flag description for special segments processing.

Note: A process belongs to the workload class if its initial thread belongs to it.

3.5.8 The svmon command flags

The same flags for **svmon** are used by the different report types, with the exception of the global and detailed segment reports.

Table 3-5 Commonly used flags of the svmon command

Flag	Description
-G	Displays a global report.
-P [*pid1... pidN*]	Displays memory usage statistics for process *pid1...pidN*. *pid* is a decimal value. If no list of process IDs (PIDs) is supplied, memory usage statistics are displayed for all active processes.
-S [*sid1...sidN*]	Displays memory-usage statistics for segments *sid1...sidN*. *sid* is a hexadecimal value. If no list of segment IDs (SIDs) is supplied, memory usage statistics are displayed for all defined segments.
-U [*lognm1...lognmN*]	Displays memory usage statistics for the login name *lognm1...lognmN*. *lognm* is a string. It is an exact login name. If no list of login identifiers is supplied, memory usage statistics are displayed for all defined login identifiers.
-C *cmd1...cmdN*	Displays memory usage statistics for the processes running the command name *cmdnm1...cmdnmN*. *cmdnm* is a string. It is the exact base name of an executable file.
-W [*clnm1...clnmN*]	Displays memory usage statistics for the workload management class *clnm1...clnmN*. *clnm* is a string. It is the exact name of a class. If no list of class names is supplied, memory usage statistics are displayed for all defined class names.
-D *sid1...sidN*	Displays memory-usage statistics for segments *sid1...sidN*, and a detailed status of all the frames of each segment.
-n	Indicates that only non-system segments are to be included in the statistics. By default, all segments are analyzed.
-s	Indicates that only system segments are to be included in the statistics. By default, all segments are analyzed.
-w	Indicates that only working segments are to be included in the statistics. By default, all segments are analyzed.

Flag	Description
-f	Indicates that only persistent segments (files) are to be included in the statistics. By default, all segments are analyzed.
-c	Indicates that only client segments are to be included in the statistics. By default, all segments are analyzed.
-u	Indicates that the objects to be printed are sorted in decreasing order by the total number of pages in real memory. It is the default sorting criteria if none of the following flags are present: -p, -g, and -v.
-p	Indicates that the object to be printed is sorted in decreasing order by the total number of pages pinned.
-g	Indicates that the object to be printed is sorted in decreasing order by the total number of pages reserved or used on paging space. This flag, in conjunction with the segment report, shifts the non-working segment at the end of the sorted list.
-v	Indicates that the object to be printed is sorted in decreasing order by the total number of pages in virtual space. This flag, in conjunction with the segment report, shifts the non-working segment to the end of the sorted list.
-b	Shows the status of the reference and modified bits of all the displayed frames (detailed report -D). Once shown, the reference bit of the frame is reset. When used with the -i flag, it detects which frames are accessed between each interval. This flag should be used with caution because of its performance impact.
-l	Shows, for each displayed segment, the list of process identifiers that use the segment and, according to the type of report, the entity name (login, command, or class) the process belongs to. For special segments, a label is displayed instead of the list of process identifiers.
System segment	This label is displayed for segments that are flagged systems.
Unused segment	This label is displayed for segments that are not used by any existing processes.
Shared library text	This label is displayed for segments that contain text of shared library, and that may be used by most of the processes (libc.a). This is to prevent the display of a long list of processes.

Flag	Description
-z	Displays the maximum memory size dynamically allocated (malloc) by **svmon** during its execution.
-m	Displays information about the source segment rather than the mapping segment when a segment is mapping a source segment.
-d	Displays the memory statistics of the processes belonging to a given entity.
-t *count*	Displays memory usage statistics for the top count object to be printed.
-i *interval* [*NumIntervals*]	Instructs the **svmon** command to print statistics out repetitively. Statistics are collected and printed every *interval* seconds. *NumIntervals* is the number of repetitions; if not specified, **svmon** runs until user interruption (Ctrl+C).

Note: If no command line flag is given, then the -G flag is implicit.

Because it may take a few seconds to collect statistics for some options, the observed interval may be larger than the specified interval.

If none of the -u, -p, -g, and -v flags are specified, -u is implicit.

3.6 The rmss command

The **rmss** command provides you with a means to simulate different sizes of real memory that are smaller than your actual machine, without having to extract and replace memory boards or reconfigure memory using logical partitions. Moreover, the **rmss** command provides a facility to run an application over a range of memory sizes, displaying, for each memory size, performance statistics such as the response time of the application and the amount of paging.

The main use for the **rmss** command is as a capacity planning tool, to determine how much memory a workload needs. It can also be used as a problem determination tool, particularly for those cases where having more memory degrades performance.

To determine whether the **rmss** command is installed and available, run the following command:

```
# lslpp -lI perfagent.tools
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
```

```
Path: /usr/lib/objrepos
  perfagent.tools           5.1.0.25  COMMITTED  Local Performance Analysis &
                                                 Control Commands

Path: /etc/objrepos
  perfagent.tools           5.1.0.25  COMMITTED  Local Performance Analysis &
                                                 Control Commands
```

Note: The memory size simulated by the **rmss** command is the total size of the machine's real memory, including the memory used by the operating system and any other programs that may be running. It is not the amount of memory used specifically by the application itself.

The **rmss** command can be invoked two ways:

1. To change the memory size and exit.
2. As a driver program that executes a specified application multiple times over a range of memory sizes and displays important statistics that describe the application's performance at each memory size.

The first invocation technique is useful when you want to get the look and feel of how your application performs at a given system memory size, when your application is too complex to be expressed as a single command, or when you want to run multiple instances of the application. The second invocation technique is appropriate when you have an application that can be invoked as an executable program or shell script file.

The **rmss** command has the following syntax:

```
rmss -c MemSize
rmss -r
rmss -p
rmss [ -d MemSize ] [ -f MemSize ] [ -n NumIterations ] [ -o OutputFile ]
[ -s MemSize ] Command
```

The commonly used flags are listed in Table 3-6.

Table 3-6 Commonly used flags of the rmss command

Flag	Description
-c *MemSize*	Changes the simulated memory size to the MemSize value, which is an integer or decimal fraction in units of megabytes.
-d *MemSize*	Specifies the increment between memory sizes to be simulated.
-f *MemSize*	Specifies the final memory size.

Flag	Description
-r	Resets the simulated memory size to the real memory size of the machine.
-s *MemSize*	Specifies the starting memory size.
-o *OutputFile*	Specifies the file into which to write the **rmss** report.

For example, to change the memory size to 500 MB, enter:

```
# rmss -c 500
Simulated memory size changed to 500 Mb.
```

To reset the memory size to the real memory size of the machine, enter:

```
# rmss -r
```

When any combination of the -s, -f, -d, -n, and -o flags is used, the **rmss** command runs as a driver program, which executes a command multiple times over a range of memory sizes, and displays statistics describing the command's performance at each memory size.

An example of the report produced by the **rmss** command follows:

```
# rmss -s 1000 -f 900 -d 10 -n 1 -o /tmp/rmss.out cp /smit.log /dev/null

Hostname:  server4.itsc.austin.ibm.com
Real memory size:  1024 Mb
Time of day:  Wed Sep 18 15:24:13 2002
Command:  cp/smit.log/dev/null

Simulated memory size initialized to 1000 Mb.

Number of iterations per memory size = 1 warmup + 1 measured = 2.

Memory size     Avg. Pageins    Avg. Response Time      Avg. Pagein Rate
(megabytes)                           (sec.)            (pageins / sec.)
--------------------------------------------------------------------------
1000                2.0               0.0                   70.2
990                 1.0               0.0                   35.4
980                 3.0               0.0                   106.6
970                 5.0               0.0                   176.7
960                 1.0               0.0                   35.0
950                 4.0               0.0                   141.0
940                 3.0               0.0                   105.9
930                 7.0               0.0                   248.6
920                 3.0               0.0                   109.7
910                 3.0               0.0                   105.0
900                 5.0               0.0                   108.9
```

```
Simulated final memory size.
```

The report consists of four columns.

- The left-most Memory Size column gives the memory size.
- The Avg. Pageins column gives the average number of page-ins that occurred when the application was run at that memory size. It is important to note that the Avg. Pageins column refers to all page-in operations, including code, data, and file reads, from all programs that completed while the application ran.
- The Avg. Response Time column gives the average amount of time it took the application to complete.
- The Avg. Pagein Rate column gives the average rate of page-ins.

3.7 The topas command

The **topas** command reports vital statistics about activity on the local system using a text-based output. It requires the AIX Version 5L (or later) perfagent.tools fileset to be installed on the system. The **topas** command has received updates since the time of writing, so check the AIX product documentation for additional features. This command is a type of compilation of diagnostic commands, such as **sar**, **vmstat**, **iostat**, and **netstat**. It allows you to see all of the statistics on one screen so it is easy to observe interactions between them. The **topas** command has the following syntax:

```
topas [ -d number_of_monitored_hot_disks ] [ -h ]
[ -i monitoring_interval_in_seconds ]
[ -n number_of_monitored_hot_network_interfaces ]
[ -p number_of_monitored_hot_processes ]
[ -w number_of_monitored_hot_WLM classes ] [ -c number_of_monitored_hot_CPUs ]
[ -P | -W ]
```

The commonly used flags are listed in Table 3-7.

Table 3-7 Commonly used flags of the topas command

Flag	Description
-i	Sets the monitoring interval in seconds. The default is 2 seconds.
-p	Specifies the number of hot processes to be monitored.
-w	Specifies the number of hot WorkLoad Manager (WLM) classes to be monitored.
-c	Specifies the number of hot CPUs to be monitored.

3.7.1 Common uses of the topas command

Figure 3-1 on page 100 shows the standard **topas** command and its output. The system host name is displayed on the top line of the screen. The line below shows the time and date as well as the sample interval used for measurement.

CPU utilization statistics

CPU utilization is graphically and numerically displayed below the date and time and is split up into a percentage of idle, wait, user, and kernel time.

Idle time The percentage of time when the processor is not performing any tasks

Wait time The percentage of time when the CPU is waiting for the response of an input output device such as a disk or network adapter

User time The percentage of time when the CPU is executing a program in user mode

Kernel time The percentage of time when the CPU is running in kernel mode

Network interface statistics

The following network statistics are available over the monitoring period.

Network The name of the interface adapter

KBPS Reports the total throughput of the interface in kilobytes per second

I-Pack Reports the number of packets received per second

O-Pack Reports the number of packets sent per second

KB-In Reports the number of kilobytes received per second

KB-Out Reports the number of kilobytes sent per second

Disk drive statistics

The following disk drive statistics are available.

Disk The name of the disk drive.

Busy% Reports the percentage of time that the disk drive was active.

KBPS Reports the total throughput of the disk in kilobytes per second. This value is the sum of KB-Read and KB-Writ.

TPS Reports the number of transfers per second or I/O requests to a disk drive.

KB-Read Reports the number of kilobytes read per second.

KB-Writ Reports the number of kilobytes written per second.

Process statistics

The top hot processes are displayed with the following headings.

Name The name of the process. Where the number of characters in the process name exceeds nine, the name will be truncated. No path name details for the process are displayed.

PID Shows the process identification number for the process. This is useful when a process needs to be stopped.

CPU% Reports on the CPU time utilized by this process.

PgSp Reports on the paging space that has been allocated to this process.

Owner Displays the owner of the process.

Event and queue statistics

This part of the report is on the right-hand side of the **topas** display screen and reports on select system global events and queues over the sampling interval.

Cswitch Reports the number of context switches per second

Syscall Reports the total number of system calls per second

Reads Reports the number of read system calls per second

Writes Reports the number of write system calls per second

Forks Reports the number of fork system calls per second

Execs Reports the number of exec system calls per second

Runqueue Reports the average number of threads that were ready to run, but were waiting for a processor to become available

Waitqueue Reports the average number of threads waiting for paging to complete

File and tty statistics

The file and tty part of the **topas** display is located on the extreme right-hand side at the top. The reported items are listed below.

Readch Reports the number of bytes read through the read system call per second

Writech Reports the number of bytes written through the write system call per second

Rawin	Reports the number of bytes read in from a tty device per second
Ttyout	Reports the number of bytes written to a tty device per second
Igets	Reports on the number of calls per second to the inode lookup routines
Namei	Reports the number of calls per second to the path lookup routine
Dirblk	Reports on the number of directory blocks scanned per second by the directory search routine

Paging statistics

There are two parts of the paging statistics reported by **topas**. The first part is total paging statistics. This simply reports the total amount of paging available on the system and the percentages free and used. The second part provides a breakdown of the paging activity. The reported items and their meanings are listed below.

Faults	Reports the number of faults
Steals	Reports the number of 4-KB pages of memory stolen by the Virtual Memory Manager per second
PgspIn	Reports the number of 4-KB pages read in from the paging space per second
PgspOut	Reports the number of 4-KB pages written to the paging space per second
PageIn	Reports the number of 4-KB pages read per second
PageOut	Reports the number of 4-KB pages written per second
Sios	Reports the number of input/output requests per second issued by the Virtual Memory Manager

Memory statistics

The memory statistics are listed below.

Real	Shows the actual physical memory of the system in megabytes
%Comp	Reports real memory allocated to computational pages
%Noncomp	Reports real memory allocated to non-computational pages
%Client	Reports on the amount of memory that is currently used to cache remotely mounted files

NFS statistics

Statistics for client and server calls per second are displayed.

WLM classes

This subsection displays a list of hot WorkLoad Manager (WLM) Classes. The maximum number of WLM classes displayed is the number of hot WLM classes being monitored as specified with the -w flag. A smaller number of classes will be displayed if other subsections are also being displayed. Pressing the w key turns off this subsection. The following fields are displayed for each class:

%CPU Utilization The average CPU utilization of the WLM class over the monitoring interval.

%Mem Utilization The average memory utilization of the WLM class over the monitoring interval.

%Blk I/O The average percent of block I/O of the WLM class over the monitoring interval. When this subsection first displays the list of hot WLM classes, the list will be sorted by the CPU% field. However, the list can be sorted by the other fields by moving the cursor to the top of the desired column.

Processes

This subsection displays a list of hot processes. The maximum number of Processes displayed is the number of hot processes being monitored as specified with the -p flag. A smaller number of processes will be displayed if other subsections are also being displayed. Pressing the p key turns off this subsection. The processes are sorted by their CPU usage over the monitoring interval. The following fields are displayed for each process:

- Name

  The name of the executable program executing in the process. The name is stripped of any path name and argument information and truncated to nine characters in length.

- Process ID

  The process ID of the process.

- %CPU Utilization

  The average CPU utilization of the process over the monitoring interval. The first time a process is shown, this value is the average CPU utilization over the lifetime of the process.

- Paging Space Used

  The size of the paging space allocated to this process. This can be considered an expression of the footprint of the process but does not include the memory used to keep the executable program and any shared libraries it may depend on.

- Process Owner (if the WLM section is off)

  The user name of the user who owns the process.

- WorkLoad Management (WLM) Class (if the WLM section is on)

  The WLM class to which the process belongs.

Note: In order to obtain a meaningful output from the **topas** command, the screen or graphics window must support a minimum of 80 characters by 24 lines. If the display is smaller than this, then parts of the output become illegible.

The output of the **topas** command is shown in Figure 3-1.

```
Topas Monitor for host:    server4.itsc.austin.ibm.com/QUEUES    FILE/TTY
Tue Sep  3 11:19:53 2002   Interval:  2         Cswitch     405  Readch      119
                                                Syscall    2863  Writech     485
Kernel    0.0   |                          |  Reads         0  Rawin         0
User      0.0   |                          |  Writes        0  Ttyout        0
Wait      0.0   |                          |  Forks         0  Igets         0
Idle    100.0   |##########################|  Execs         0  Namei         0
                                                Runqueue    1.0  Dirblk        0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out  Waitqueue   0.0
lo0       2.2     0.4     0.4     1.1     1.1
en0       0.4     1.9     0.4     0.1     0.3  PAGING           MEMORY
                                                Faults         0  Real,MB    1023
Disk     Busy%     KBPS     TPS KB-Read KB-Writ  Steals         0  % Comp     25.6
hdisk4    0.0      0.0     0.0     0.0     0.0  PgspIn         0  % Noncomp  10.2
hdisk0    0.0      0.0     0.0     0.0     0.0  PgspOut        0  % Client    0.5
                                                PageIn         0
WLM-Class (Active)      CPU%    Mem%  Disk-I/O%  PageOut        0  PAGING SPACE
                           0       0          0  Sios           0  Size,MB     512
                                                                  % Used      1.3
Name            PID CPU% PgSp Class             NFS (calls/sec)   % Free     98.6
topas         29028  0.1  1.8 System            ServerV2       0
xterm         31558  0.0  1.9 System            ClientV2       0   Press:
xmperf        19392  0.0  3.0 System            ServerV3       0   "h" for help
wlmsched       2838  0.0  0.0 System            ClientV3       0   "q" to quit
```

Figure 3-1 The topas command output

In AIX 5L Version 5.1, the **topas** program has been enhanced to add two alternate screens, the CPU utilization report has become an optional subsection, and the fixed section includes an additional subsection with NFS statistics, while

the variable section includes a new WLM subsection. The first alternate screen (reachable with the **P** command or the -P flag), presents the list of busiest processes, similar to the processes subsection of the main screen, but with more columns. This screen is sortable by any of its columns. The second alternate screen (reachable with the **W** command, or the -W flag), is divided into two sections. The top section is the same list of busiest WLM classes as presented in the WLM subsection of the main screen, also sortable by any of its columns. When the user selects one the WLM classes shown using the arrow keys and the f key, the second section of the screen will show the list of hot processes within the selected WLM class.

While **topas** is running, it accepts one-character subcommands. Each time the monitoring interval elapses, the program checks for one of the following subcommands and responds to the action requested:

a The a key shows all of the variable subsections being monitored (CPU, network, disk, WLM, and process). Pressing the a key always returns the topas command to the initial main display.

c The c key toggles the CPU subsection between the cumulative report, off, and a list of the busiest CPUs. The number of busiest CPUs displayed will depend upon the space available on the screen.

d The d key toggles the disk subsection between a list of busiest disks, off, and the report on the total disk activity of the system. The number of busiest disks displayed will depend upon the space available on the screen.

h The h key shows the help screen.

n The n key toggles the network interfaces subsection between a list of busiest interfaces, off, and the report on the total network activity of the system. The number of busiest interfaces displayed will depend upon the space available on the screen.

w The w key toggles the WorkLoad Management (WLM) classes subsection on and off. The number of busiest WLM classes displayed will depend upon the space available on the screen.

p The p key toggles the hot processes subsection on and off. The number of busiest processes displayed will depend upon the space available on the screen.

P The uppercase P key replaces the default display with the full-screen process display. This display provides more detailed information about processes running on the system than the process section of the main display. When the P key is pressed again, it toggles back to the default main display.

W The uppercase W key replaces the default display with the full-screen WLM class display. This display provides more detailed information about WLM classes and processes assigned to classes. When the W key is pressed again, it toggles back to the default main display.

f Moving the cursor over a WLM class and pressing the f key displays the list of top processes in the class at the bottom of the WLM screen. This key is valid only when **topas** is in the full-screen WLM display (by using the W key or the -W flag).

q Quits the program.

r Refreshes the display.

Check the latest documentation for a full set of subcommands and command line options.

3.8 The emstat command

The PowerPC architecture no longer supports, in hardware, 35 POWER instructions. To maintain compatibility with older binaries (which may contain these deleted instructions), the AIX Version 4 kernel includes emulation routines that provide support for the deleted instructions. Attempting to execute a deleted instruction results in an illegal instruction exception. The kernel decodes the illegal instruction, and, if it is a deleted instruction, the kernel runs an emulation routine that functionally emulates the instruction.

The **emstat** command reports statistics regarding how many instructions the system must emulate. The emulated instruction count should be used to determine if an application needs to be recompiled to eliminate instructions that must be emulated on 601 PowerPC, 604 PowerPC, RS64, or other non-POWER processors. If an instruction must be emulated, more CPU cycles are required to execute this instruction than an instruction that does not have to be emulated.

Most emulation problems are seen on older PowerPC 604 systems. A typical example is a PowerPC 601 system that is upgraded to a 604 system. If performance slows down instead of improving, it is most likely due to emulation.

The solution to emulation is to recompile the application in common mode. The default architecture platform for compilations on AIX Version 4 is common architecture. However, the default architecture on AIX Version 3 was for POWER, POWER2, and PowerPC 601. If these binaries ran on a PowerPC 604, some instructions may be emulated.

To determine whether the **emstat** command is installed and available, run the following command:

```
# lslpp -l bos.adt.samples
```

Note: In AIX 5L, the **emstat** command is located in the perfagent.tools fileset.

The **emstat** command works similarly to the **vmstat** command in that you specify an interval time in seconds, and, optionally, the number of intervals. The value in the first column is the cumulative count since system boot, while the value in the second column is the number of instructions emulated during the specified interval. Emulations on the order of many thousands per second can have an impact on performance:

```
# /usr/samples/kernel/emstat 2
emstat total count      emstat interval count
           965                       965
           965                         0
           965                         0
           965                         0
           965                         0
           967                         2
           967                         0
           967                         0
           974                         7
           974                         0
           974                         0
           974                         0
           974                         0
           974                         0
          1284                       310
          2171                       887
          3325                      1154
```

Once emulation has been detected, the next step is to determine which application is emulating instructions. This is much harder to determine. One way is to run only one application at a time and monitor it with the **emstat** program.

3.9 The /proc file system

AIX 5L provides support of the /proc file system. This pseudo file system maps processes and kernel data structures to corresponding files.

The output of the **mount** and **df** commands showing /proc is provided in the following examples:

```
# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    Sep 11 16:52 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    Sep 11 16:52 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    Sep 11 16:52 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    Sep 11 16:52 rw,log=/dev/hd8
         /dev/hd1         /home            jfs    Sep 11 16:53 rw,log=/dev/hd8
         /proc            /proc            procfs Sep 11 16:53 rw

# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           65536     27760   58%     2239    14% /
/dev/hd2         1507328    242872   84%    22437    12% /usr
/dev/hd9var        32768     16432   50%      448    11% /var
/dev/hd3          557056    538008    4%      103     1% /tmp
/dev/hd1           32768     31608    4%       47     2% /home
/proc                  -         -    -        -     -  /proc
```

The entry in the /etc/vfs file appears as follows:

```
# lsvfs procfs
procfs  6       none    none
```

Each process is assigned a directory entry in the /proc file system with a name identical to its process ID. In this directory, several files and subdirectories are created corresponding to internal process control data structures. Most of these files are read-only, but some of them can also be written to and be used for process control purposes. The interfaces to these files are the standard C language subroutines open(), read(), write(), and close(). It is possible to have several concurrent readers, but for reliability reasons, the first write access should use the exclusive flag, so that subsequent opens for write access fail. The description of the data structures used can be found in /usr/include/sys/procfs.h. The ownership of the files in the /proc file system is the same as for the processes they represent. Therefore, regular users can only access /proc files that belong to their own processes.

A simple example illustrates this further. Suppose a process is waiting for standard input (the information in the process data structures is basically static). If you look at an active process, a lot of the information would constantly change:

```
# ls -l /proc/19082/
total 0
dr-xr-xr-x   1 root     system            0 Sep 15 15:12 .
dr-xr-xr-x   1 root     system            0 Sep 15 15:12 ..
-rw-------   1 root     system            0 Sep 15 15:12 as
```

```
-r--------   1 root     system          128 Sep 15 15:12 cred
--w-------   1 root     system            0 Sep 15 15:12 ctl
dr-xr-xr-x   1 root     system            0 Sep 15 15:12 lwp
-r--------   1 root     system            0 Sep 15 15:12 map
dr-x------   1 root     system            0 Sep 15 15:12 object
-r--r--r--   1 root     system          448 Sep 15 15:12 psinfo
-r--------   1 root     system         1024 Sep 15 15:12 sigact
-r--------   1 root     system         1520 Sep 15 15:12 status
-r--r--r--   1 root     system            0 Sep 15 15:12 sysent
```

Table 3-8 provides the function of the pseudo files listed in the previous output.

Table 3-8 Function of pseudo files in /proc/<pid> directory

Pseudo file name	Function
as	Read/write access to address space
cred	Credentials
ctl	Write access to control process, for example, stop or resume
lwp directory	Kernel thread information
map	Virtual address map
object directory	Map file names
psinfo	Information for the **ps** command; readable by everyone
sigact	Signal status
status	Process state information, such as address, size of heap, or stack
sysent	Information about system calls

The pseudo file named *as* allows you to access the address space of the process, and as it can be seen by the rw (read/write) access flags, you can read and write to the memory belonging to the process.

It should be understood that only the user regions of the process' address can be written to under /proc. Also, a copy of the address space of the process is made while tracing under /proc. This is the address space that can be modified. This is done so when the as file is closed; the original address space is unmodified.

The cred file provides information about the credentials associated with this process. Writing to the ctl file allows you to control the process; for example, to stop or to resume it. The map file allows access to the virtual address map of the process. Information usually shown by the **ps** command can be found in the psinfo file, which is readable for all system users. The current status of all signals

associated with this process is recorded in the sigact file. State information for this process, such as the address and size of the process heap and stack (among others), can be found in the status file. Finally, the sysent file allows you to check for the system calls available to this process.

The object directory contains files with names as they appear in the map file. These files correspond to files mapped in the address space of the process. For example, the content of this directory appears as follows:

```
# ls -l /proc/19082/object
total 13192
dr-x------   1 root     system            0 Sep 15 15:09 .
dr-xr-xr-x   1 root     system            0 Sep 15 15:09 ..
-r-xr-xr-x   1 bin      bin            6264 Aug 24 21:16 a.out
-rwxr-xr-x   1 bin      bin           14342 Aug 22 22:37 jfs.10.5.10592
-r-xr-xr-x   2 bin      bin         6209308 Aug 24 13:03 jfs.10.5.2066
-r--r--r--   1 bin      bin          118267 Aug 24 15:06 jfs.10.5.2076
-r-xr-xr-x   1 bin      bin           11009 Aug 24 14:59 jfs.10.5.4129
-r--r--r--   1 bin      bin          377400 Aug 24 15:05 jfs.10.5.4161
-r-xr-xr-x   1 bin      bin            6264 Aug 24 21:16 jfs.10.5.6371
```

The a.out file always represents the executable binary file for the program running in the process itself. Because the example program is written in C and must use the C runtime library, it can be concluded from the size of the entry named jfs.10.5.2066 that this corresponds to the /usr/ccs/lib/libc.a file. Checking this file reveals that the numbers in the file name are the major and minor device numbers, and the inode number, respectively. This can be seen in the following output, where /usr corresponds to /dev/hd2 and the **ncheck** command is used to find a file belonging to an inode in a specific file system:

```
# ls -l /dev/hd2
brw-rw----   1 root     system    10,  5 Sep 20 16:09 /dev/hd2
# ncheck -i 2066 /dev/hd2
/dev/hd2:
2066    /ccs/lib/libc.a
```

The lwp directory has subdirectory entries for each kernel thread running in the process. The term *lwp* stands for lightweight process and is the same as the term *thread* used in the AIX documentation. It is used in the context of the /proc file system to keep a common terminology with the /proc implementation of other operating systems. The names of the subdirectories are the thread IDs. The test program has only one thread with the ID 54891, as shown in the output of the **ps** command. Therefore, only the content of this one thread directory is shown:

```
# ps -mo THREAD -p 19082
    USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
    root 19082 20678      - A    0  83  1 700e6244   200001  pts/3   - wc
       -     -     -  54891 S    0  83  1 700e6244    10400      -   - -
# ls -l /proc/19082/lwp/54891
```

```
total 0
dr-xr-xr-x   1 root     system            0 Sep 15 15:03 .
dr-xr-xr-x   1 root     system            0 Sep 15 15:03 ..
--w-------   1 root     system            0 Sep 15 15:03 lwpctl
-r--r--r--   1 root     system          120 Sep 15 15:03 lwpsinfo
-r--------   1 root     system         1200 Sep 15 15:03 lwpstatus
```

The lwpctl, lwpsinfo, and lwpstatus files contain thread-specific information to control this thread, for the **ps** command, and about the state, similar to the corresponding files in the /proc/*pid* directory.

As an example of what can be obtained from reading these files, the following lines show the content of the cred file (after the use of the **od** command):

```
# ls -l /proc/19082/cred
-r--------   1 root     system          128 Sep 15 15:07 /proc/19082/cred
# od -x /proc/19082/cred
0000000  0000 0000 0000 0000 0000 0000 0000 0000
*
0000160  0000 0000 0000 0007 0000 0000 0000 0000
0000200  0000 0000 0000 0002 0000 0000 0000 0003
0000220  0000 0000 0000 0007 0000 0000 0000 0008
0000240  0000 0000 0000 000a 0000 0000 0000 000b
0000260
```

The output shows, in the leftmost column, the byte offset of the file in octal representation. The remainder of the lines are the actual content of the file in hexadecimal notation. Even if the directory listing shows the size of the file to be 128 bytes or 0200 bytes in octal, the actual output is 0260 or 176 bytes in size. This is due to the dynamic behavior of the last field in the corresponding structure. The digit 7 in the line with the number 0160 specifies the number of groups the user ID running this process belongs to. Because every user ID is at least part of its primary group, but belongs possibly to a number of other groups that cannot be known in advance, only space for the primary group is reserved in the cred data structure. In this case, the primary group ID is zero because the user ID running this process is root. Reading the complete content of the file, nevertheless, reveals all the other group IDs the user currently belongs to. The group IDs in this case (2, 3, 7, 8, 0xa (10), and 0xb (11)) map to the groups bin, sys, security, cron, audit, and lp. This is exactly the set of groups the user ID root belongs to by default.

3.10 General performance guidelines

Table 3-9 on page 108 lists some general rule-of-thumb formula for investigating system performance.

Table 3-9 General performance guidlines

Formula	Command	Description
%usr + %sys < 80%	**sar** **vmstat**	If %usr and %sys are greater than 80 percent, then the system may be CPU bound.
po / fr > 1/h	**vmstat**	If the ratio of po:fr is greater than 1–6, then the system may be thrashing.
fr / sr = high	**vmstat**	If the ratio of fr:sr is high, then the system may be over committing memory.
%user + %sys > 70%	**iostat**	If utilization is greater than 70 percent processes are waiting longer than necessary for I/O to complete, as most processes sleep while waiting for their I/O request to complete.

3.11 Quiz

The following assessment questions help verify your understanding of the topics discussed in this chapter.

1. When monitoring a system with **vmstat**, which of the following values under the cpu heading would indicate that the system is probably CPU bound?
   A. The value of wa is greater than 70.
   B. The value of id is greater than 70.
   C. The sum of us and sy is consistently between 99 and 100.
   D. The sum of id and wa is consistently between 99 and 100.
2. Which of the following commands should be used to show the number of system calls per second that are being executed?
   A. **pstat**
   B. **vmstat**
   C. **filemon**
   D. **lockstat**
3. When using **vmstat** with intervals, which of the following describes the values of the first line of statistics?
   A. Averages since the system boot
   B. Averages since the top of the hour
   C. Averages since the beginning of the day
   D. Averages since the beginning of the month

4. Using the report from **vmstat -s**, as shown in the exhibit, which of the following indicates how the paging statistics can be improved?

```
vmstat -s
   11228009 total address trans. faults
    1791982 page ins
     419439 page outs
     221142 paging space page ins
     231045 paging space page outs
          0 total reclaims
    3461761 zero filled pages faults
      10630 executable filled pages faults
    3298154 pages examined by clock
         52 revolutions of the clock hand
    1199652 pages freed by the clock
      76372 backtracks
          0 lock misses
          0 free frame waits
          0 extend XPT waits
     254265 pending I/O waits
     823719 start I/Os
     823719 iodones
   43493481 cpu context switches
  140637006 device interrupts
          0 software interrupts
          0 traps
   85003313 syscalls
```

   A. Add paging space.

   B. Increase CPU power.

   C. Add physical memory.

   D. Reduce the number of system calls.

5. When monitoring a system using **vmstat** with an interval, which of the following conclusions should be drawn about the metrics under page pi and page po?

   A. Occasional small numbers are normal.

   B. Any non-zero value of pi or po indicates a RAM shortage.

   C. Any non-zero value of pi or po indicates a paging space shortage.

   D. Large values are normal only on multi-processor systems.

6. When monitoring a system using **/usr/bin/vmstat** with an interval, how does the size of the interval affect the page pi and page po metrics?

   A. It should not; the metric is in pages per second.

   B. The larger the interval, the larger the values.

   C. The interval only affects the first data line of output.

   D. The interval only affects the pi column, not the po column.

7. Which of the following commands will display the total number of pages paged out to paging space since the system was booted?

   A. **lsps**

   B. **iostat**

   C. **uptime**

   D. **/usr/bin/vmstat -s**

8. Which of the following tools should be used to analyze memory usage for the whole system at a point in time?

   A. **iostat**

   B. **svmon**

   C. **netstat**

   D. **filemon**

9. In order to obtain a list of the top memory users on an AIX Version 4 system, which of the following commands should be used?

   A. **pstat**

   B. **tprof**

   C. **svmon**

   D. **filemon**

10. Using the **ps -el** output, as shown in the exhibit, which of the following conclusions is most appropriate to draw?

```
F      S UID PID  PPID C   PRI NI ADDR SZ   WCAN    TTY    TIME   CMD
202803 S 0   1    0    0   60  20 1004 528            -   131:33 init
260801 S 0   1406 1    0   60  20 4550 208            -   0:00   srcmstr
240801 S 0   1694 1    0   60  20 37cd 176  5a6a024   -   0:01   cron
260801 S 0   2448 1    0   60  20 5d57 144            -   3:18   portmap
240801 S 0   2836 1    0   60  20 34cd  72  5e34398   -   42:42  syncd
```

```
F      S UID PID  PPID C   PRI NI ADDR SZ  WCAN    TTY    TIME  CMD
 42801 S 0   3606 1    0   60  20 50d4 284 cc98       -   0:01  errdemon
260801 S 0   5255 1    0   60  20 5535 148             -   1:00  syslogd
240801 S 0   5541 1    0   59  20 7d9f  60 3f2f8       -   0:00  llbd
240801 S 200 5750 79989  0   60  20 a83  180          pts/10 0:00  -ksh
260801 S 0   6040 1    0   60  20 1565 468             -   7:53  snmpd
 60801 S 0   6299 1    0   60  20 2d6b 224             -   0:00  x_st_mgr
240801 S 0   6502 14060  0   60  20 14e7 184 12d0440     -   0:05  qdaemon
 40001 S 0   6659 1    0   23    -- 3aae 312 1fca54   hft/0 0:32  userprog
```

A. The snmpd process is running at a fixed priority of 60.

B. If both the syncd process and the portmap process become runable at the same time, then the syncd process would get scheduled first.

C. If both the userprog process and the qdaemon process become runable at the same time, then the qdaemon process would get scheduled first.

D. If both the syslogd process and the llbd process become runable at the same time, then the llbd process would get scheduled first.

11. Which of the following general operation techniques will best provide a measurable performance improvement of a system?

A. Attach SCSI devices in ID order.

B. Perform backups at regular intervals.

C. Perform regular preventative maintenance of devices with moving parts.

D. Reschedule the processing to balance the workload.

12. A developer tried to run a **sar** report and received the following error:

```
sar:0551-201 cannot open /usr/adm/sa/sa12
```

Which of the following procedures should be performed so that the developer can obtain **sar** reports?

A. Run bosboot -a -L**.**

B. Edit inittab to collect data.

C. Edit crontab to start collecting data.

D. Run **bosboot -ad /dev/ipldevice** and then reboot.

13. After a migration from an older POWER server to a PowerPC-based server, users complain about decreased performance. Which of the following tools should be used to investigate the performance problem?

A. **netstat**

B. **iostat**

C. **perfpmr**

D. **emstat**

14. To examine the Exhibit, press the Exhibit button. Which of the following conclusions is most appropriate to be drawn from looking at the **topas** output?

A. The system is an AIX 5.x and WLM has been installed.

B. The system is an AIX 4.x and WLM has been installed.

C. From the command line, **topas -d2 -i4 -n2 -p20 -w2 -c2** was run on an AIX 5.x system.

D. From the command line, **topas -d2 -i2 -n2 -p20 -w2 -c2** was run on an AIX 4.x system.

15. A system administrator wishes to display the five most active processes and the two most active WLM classes but no disk information every 5 seconds. Which of the following commands would accomplish this?

A. **topas -i5 -w0 -p0**

B. **topas -i5 -p5 -n0 -d0 -w0**

C. **topas -i5 -p5 -d0**

16. Which of the following is *true* about the page-in and page-out metrics produced by **vmstat -s** output?

    A. Only file space for page-ins and page-outs is included.

    B. Only client paging for page-ins and page-outs is included.

    C. Only paging space for page-ins and page-outs is included.

    D. Paging to both the file space and paging space is included.

17. Which of the following is *true* of **vmstat** on AIX 5L?

    A. The re column always shows 0.

    B. The b column is always 0 for SMP machines.

    C. The sum of us, sys, id, and wa is always 100.

    D. The sum of r and b is equal to the number of CPUs times two.

18. When monitoring a system using **vmstat** with an interval, which of the following conclusions should be drawn about the metrics under page pi and page po?

    A. If pi is twice the value of po then more RAM is needed.

    B. If po is zero and pi is high it does not indicate a memory shortage.

    C. If po is zero and pi is high there is not enough paging on the system.

    D. The pi/po ratio should never be greater than 1 unless both values are 0.

19. Which of the following options from **vmstat** indicates that a system is thrashing?

    A. po/fr > 1/h

    B. (fr/sr) > re

    C. (maxperm/minperm) > po

    D. re > 0

20. What information does the line jfs.10.5.2002 provide in the following example?

```
/proc/9808/object>#ls -l
total 0
dr-xr-xr-x   1 root     printq    Oct 5 10:34 .
dr-xr-xr-x   1 root     system    Oct 5 10:34 ..
-r-xr-xr-x   1 bin      bin       Oct 5 10:34  a.out
-rwxr-xr-x   1 bin      bin       Oct 5 10:34 jfs.10.5.2002
```

    A. The first two numbers are the major/minor numbers and the 2002 is the inode number for the process.

    B. The first two numbers are date related and the 2002 is the process ID.

C. The **ncheck -s** command is run against the jfs.10.5.2002 file to determine what the inode is for the process.

D. An od -x /proc/9808/object/jfs.10.5.2002 will give me the primary group, user ID, and current size of the running process.

21. Which of the following commands would a system administrator run to view memory usage by segment?

A. **svmon**

B. **tprof**

C. **dbx**

D. **bootinfo**

22. To get a general idea of memory utilization of a system at the current moment, which of the following commands should be used?

A. **ps aux**

B. **lsps -s**

C. **bootinfo -r**

D. **iostat 5 5**

23. Which of the following performance tools will provide source statement profiling and a summary of all CPU usage?

A. **prof**

B. **gprof**

C. **tprof**

D. **kprof**

3.11.1 Answers

The following are the preferred answers to the questions provided in this section.

1. C
2. B
3. A
4. C
5. A
6. A
7. D
8. B
9. C
10. D
11. D
12. C
13. D
14. A
15. C
16. D
17. A
18. B
19. A
20. A
21. A
22. A
23. C

3.12 Exercises

The following exercises provide sample topics for self study. They will help ensure comprehension of this chapter.

1. Describe how the **sar** command is set up to collect historical data.
2. Describe the differences between the **sa1** and **sa2** commands.

3. Describe how the **crontab** is modified to allow **sar** to collect data.
4. When running the **vmstat** command, there are either five or six measurement columns; describe all five or six and show the flag required for the sixth column.
5. Describe how a system could be described as CPU bound using the **vmstat** command. Also, what are the percentages for single- and multi-processor systems?
6. Name the seven types of reports the **svmon** command creates and give a brief description of each.
7. Which **svmon** command flags are used to display a programs resource utilization?
8. Which **svmon** command flags are used to display a user resource utilization on the system?

4

Disk I/O performance monitoring tools

The following topics are discussed in this chapter:

- Logical Volume Manager (LVM) performance analysis, using the **lslv** command
- Journaled file system (JFS) performance analysis tools, using the **filemon** and **fileplace** commands

All topics and tools discussed in this chapter will provide you with a set of methods for determining logical volume manager, file system, and disk I/O related performance issues.

4.1 Overview

With the AIX operating system, the handling of disk-related I/O is based upon different functional levels, as shown in Figure 4-1.

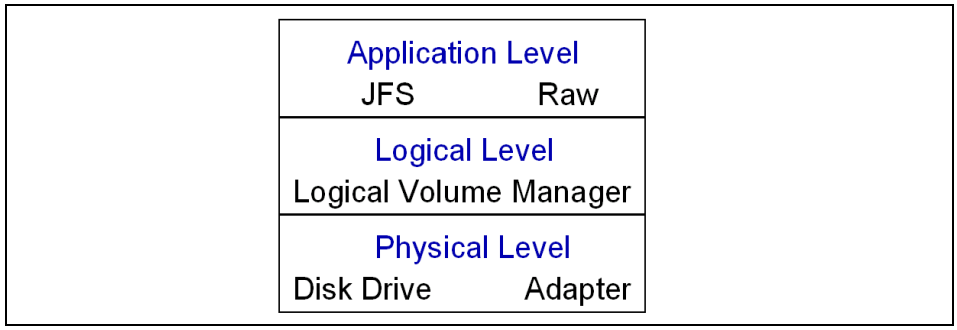


Figure 4-1 Disk, LVM, and file system levels

The lowest level is the physical level and consists of device drivers accessing the physical disks and using the corresponding adapters. The next level is the logical level, managed by the Logical Volume Manager (LVM), which controls the physical disk resources. The LVM provides a logical mapping of disk resources to the application level. The application level can consist of either the journaled file system (JFS) or raw access, for example, used by relational database systems.

The performance analysis tools discussed in the following section focus on the logical level (LVM) and on the application level (JFS). The monitoring of the physical level is primarily done by using the **iostat** command, which is described in 4.2, “The iostat command” on page 118.

Covering the entire subject of the AIX Logical Volume Manager is beyond the scope of this publication. For more background information on the AIX Logical Volume Manager, refer to *AIX Logical Volume Manager From A to Z: Introduction and Concepts*, SG24-5432, as well as the *AIX Version System Management Guide: Operating System and Devices*, SC23-2525.

4.2 The iostat command

The **iostat** command is a useful tool that provides a first indication of I/O-related performance problems. The **iostat** command is capable of reporting CPU statistics, terminal I/O statistics, and disk I/O statistics, which help identify the I/O load on individual components, such as hard disks.

The information from **iostat** reports can be used to modify system configurations to improve I/O load distribution between physical disks. The **iostat** command extracts data from AIX kernel I/O counters in the kernel address space, which are updated at every clock tick (1/100 second) for TTY as well as CPU and I/O subsystem activity.

The syntax of the **iostat** command is as follows:

```
iostat [-d | -t] [physicalVolume ...] [interval [count]]
```

The commonly used flags are provided in Table 4-1.

Table 4-1 Commonly used flags of the iostat command

Flag	Description
-d	This flag displays only the disk utilization report. The -d flag is exclusive of the -t flag.
-t	This flag displays only the TTY and CPU usage. The -t flag is exclusive of the -d flag.

By using the physicalVolume parameter (by specifying the physical volume (PV) name of the individual disk or CD-ROM), **iostat** generates an I/O report only for the specified PVs. If no PVs are specified, the **iostat** command generates a report for all drives.

The interval parameter specifies the amount of time in seconds between each report. The count parameter specifies the number of I/O reports generated. If the interval parameter is specified without the count parameter, the **iostat** command generates reports continuously.

An example of the **iostat** command is as follows:

```
# iostat 1 2

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0         41.4              61.1      0.1       38.9       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk3           0.0       0.3       0.0     258032    224266
hdisk2           0.1       1.1       0.0     258088   1658678
hdisk0           0.0       0.9       0.1     746152    725871
hdisk1           0.1       1.5       0.0     974788   1660027
cd0              0.0       0.0       0.0          0         0
hdisk4           0.0       0.2       0.0     323080     40480
```

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        603.5              91.5      8.5       0.0       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk3          16.0     2809.0     117.7       2816         0
hdisk2          16.0     2868.8     122.7       2876         0
hdisk0          91.8     8419.0     263.3          0      8440
hdisk1          21.9     2820.9     117.7       2828         0
cd0              0.0        0.0       0.0          0         0
hdisk4           0.0        0.0       0.0          0         0
```

This example shows the output of an **iostat** command run that is updated every second (interval=1) and generated only two reports (count = 2).

Each report is a combination of a TTY and CPU utilization report and a disk utilization report. This example shows a system with five hard disks (hdisk0–hdisk4) and one CD-ROM drive. The first report shows the summary statistics since system startup, providing the collective summary of I/O operations on each drive. From the previous example, you can determine that hdisk1 has been the most actively used drive.

The second report provided is the actual drive usage during a one second interval.

During the report, there was a **copy** command started to provide disk I/O activity.

4.2.1 Historical disk I/O

In AIX 5L Version 5.1 and later, the system does not collect a history of disk activity by default, as some system resources are consumed for this operation. The system administrator has to decide whether to activate the disk I/O history or not.

If the disk I/O history is disabled, the **iostat** command displays a message similar to the following:

```
# iostat 1 1

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0         41.5              61.2      0.1       38.8       0.0
                " Disk history since boot not available. "
```

Collecting disk I/O history is an AIX operating system setting that can be enabled or disabled in SMIT using **smit chgsys**. Figure 4-2 on page 121 shows the corresponding SMIT screen.

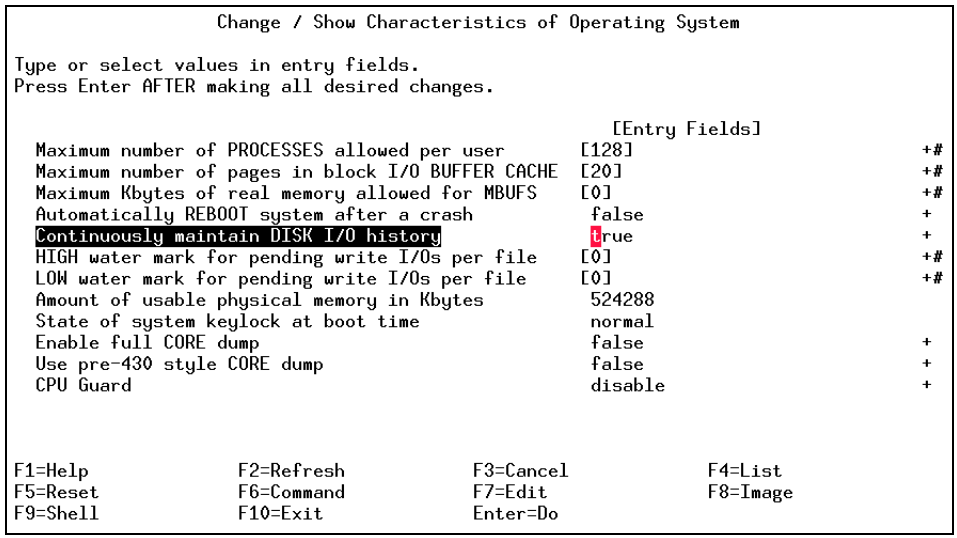


Figure 4-2 SMIT screen for changing characteristics of operating system

When the historical disk I/O is activated, ignore the first report if you are looking for real-time behavior of your system.

4.2.2 Using disk I/O pacing

Disk-I/O pacing is intended to prevent programs that generate very large amounts of output from saturating the system's I/O facilities and causing the response times of less-demanding programs to deteriorate. Disk-I/O pacing enforces per-segment (which effectively means per-file) high and low water marks on the sum of all pending I/Os. When a process tries to write to a file that already has high water mark pending writes, the process is put to sleep until enough I/Os have completed to make the number of pending writes less than or equal to the low water mark. The logic of I/O-request handling does not change. The output from high-volume processes is slowed down somewhat.

The high and low water marks are set with the SMIT command by selecting **System Environments -> Change/Show Characteristics of Operating System**, as in Figure 4-3 on page 120, and then entering the number of pages for the high and low water marks. The default value for the high and low water marks is 0, which disables pacing. Changes to the maxpout and minpout values take effect immediately and remain in place until they are explicitly changed.

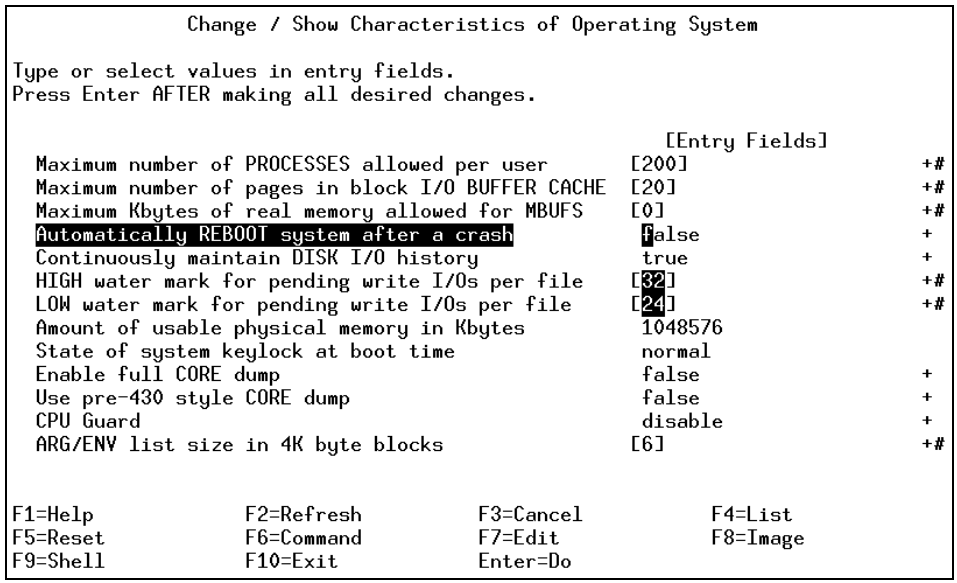


Figure 4-3 The smitty chgsys fastpath to set high and low water marks

The effect of pacing on performance can be demonstrated with an experiment that consists of starting a **vi** editor session on a new file while another process is copying a 64 MB file with the **cp** command. The file is copied from hdisk1 to hdisk0 and the **vi** executable program is located on hdisk0. For the **vi** session to start, it must page itself in as well as perform a few other I/Os, which it does randomly one page at a time. This takes about 50 physical I/Os, which can be completed in 0.71 seconds on a slow machine when there is no contention for the disk. With the high water mark set to the default of 0, the logical writes from the **cp** command run ahead of the physical writes, and a large queue builds up.

Each I/O started by the **vi** session must wait its turn in the queue before the next I/O can be issued, and thus the **vi** command is not able to complete its needed I/O until after the **cp** command finishes. Table 4-2 shows the elapsed seconds for **cp** execution and **vi** initialization with different pacing parameters.

Table 4-2 I/O pacing parameters effect on the cp and vi commands

High water mark	Low water mark	cp (sec)	vi (sec)
0	0	50	**vi** not done
0	0	50	**vi** finished after **cp**
9	6	77	2.7
17	8	64	3.4

High water mark	Low water mark	cp (sec)	vi (sec)
17	12	58	3.6
33	16	55	4.9
33	24	52	9.0

It is important to notice that the **cp** duration is always longer when pacing is set. Pacing sacrifices some throughput on I/O-intensive programs to improve the response time of other programs. The challenge for a system administrator is to choose settings that result in a throughput/response-time trade-off that is consistent with the organization's priorities.

The high and low water marks were chosen by trial and error, based on our knowledge of the I/O path. Choosing them is not straightforward because of the combination of write-behind and asynchronous writes. High water marks of (4 * *n*) + 1, where *n* is a positive integer, work particularly well because of the following interaction:

- The write-behind feature sends the previous four pages to disk when a logical write occurs to the first byte of the fifth page.
- If the pacing high water mark were a multiple of 4 (for example, 8), a process would hit the high water mark when it requested a write that extended into the ninth page. It would then be put to sleep before the write-behind algorithm had an opportunity to detect that the fourth dirty page is complete and the four pages were ready to be written.
- The process would then sleep with four full pages of output until its outstanding writes fell below the pacing low water mark.
- If, on the other hand, the high water mark had been set to 9, write-behind would get to schedule the four pages for output before the process was suspended.

One limitation of pacing is that it does not offer as much control when a process writes buffers larger than 4 KB. When a write is sent to the VMM and the high water mark has not been met, the VMM performs start I/Os on all pages in the buffer, even if that results in exceeding the high water mark. Pacing works well on the **cp** command because the **cp** command writes 4 KB at a time; but if the **cp** command wrote larger buffers, the times shown in Table 4-2 on page 122 for starting the **vi** session would increase.

Disk-I/O pacing is a tuning parameter that can improve interactive response time in some situations where foreground or background programs that write large volumes of data are interfering with foreground requests. If not used properly, however, it can reduce throughput excessively. The settings in the previous table

are a good place to start, but some experimenting will be needed to find the best settings for your workload. For the newer systems, view these numbers as the minimum starting point.

Programs whose presence in a workload may make imposition of disk-I/O pacing necessary include:

- Programs that generate large amounts of output algorithmically, and thus are not constrained by the time required to read input. Some such programs may need pacing on comparatively fast processors and not need it on comparatively slow processors.
- Programs that write large, possibly somewhat modified, files that have been read in their entirety shortly before writing begins (by a previous command, for example).
- Filters, such as the **tar** command, that read a file and write it out again with little processing. The need for pacing can be exacerbated if the input is being read from a faster disk drive than the output is being written to.

Setting automatic reboot

Set the server to automatically reboot after a system crash with the SMIT command by selecting **System Environments -> Change/Show Characteristics of Operating System**, as in Figure 4-4 on page 123.

```
                Change / Show Characteristics of Operating System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Maximum number of PROCESSES allowed per user       [200]                +#
  Maximum number of pages in block I/O BUFFER CACHE  [20]                 +#
  Maximum Kbytes of real memory allowed for MBUFS    [0]                  +#
  Automatically REBOOT system after a crash           true                +
  Continuously maintain DISK I/O history              true                +
  HIGH water mark for pending write I/Os per file    [0]                  +#
  LOW water mark for pending write I/Os per file     [0]                  +#
  Amount of usable physical memory in Kbytes          1048576
  State of system keylock at boot time                normal
  Enable full CORE dump                               false               +
  Use pre-430 style CORE dump                         false               +
  CPU Guard                                           disable             +
  ARG/ENV list size in 4K byte blocks                [6]                  +#


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 4-4 The smitty chgsys fastpath to set automatic reboot

4.2.3 TTY and CPU utilization report

The first report section displayed by the **iostat** command contains the TTY and CPU utilization report.

The following columns are displayed and their meanings provided:

tin Shows the total characters per second read by all TTY devices

tout Indicates the total characters per second written to all TTY devices

% user Shows the percentage of CPU utilization that occurred while executing at the user level (application)

% sys Shows the percentage of CPU utilization that occurred while executing at the system level (kernel)

% idle Shows the percentage of time that the CPU or CPUs were idle while the system did not have an outstanding disk I/O request

% iowait Shows the percentage of time that the CPU or CPUs was idle during which the system had an outstanding disk I/O request

The TTY information columns, tin and tout, show the number of characters read and written by all TTY devices. This includes both real and pseudo TTY devices. Real TTY devices are those connected to an asynchronous port, such as serial terminals, modems, FAXes, and so on. Pseudo TTY devices are **telnet** sessions and **xterm** (or other X-based terminal emulators, such as **dtterm** and **aixterm**).

Since the processing of character I/O consumes CPU resources, it is important to monitor the relation between increased TTY activity and CPU utilization. If such a relationship exists, the TTY devices, along with the applications using these TTY devices, should be analyzed. For example, a FAX application could be improved by enhancing the speed of the TTY port parameters so that a file transfer would become faster and more efficient.

The CPU statistics columns % user, % sys, % idle, and % iowait provide information about the CPU usage. The same information is also reported in the **vmstat** command output in the columns us, sy, id, and wa.

In general, a high % iowait indicates that the system has a memory shortage due to paging or an inefficient I/O subsystem configuration. Understanding the I/O bottleneck and improving the efficiency of the I/O subsystem requires more data than **iostat** can provide.

On a system that shows 0% iowait when running the **iostat** command, there still might be an I/O bottleneck if the I/O to the disks is running at full capacity, as this will not show up in the % iowait column.

When the **iostat** command report shows that a CPU-bound situation does not exist with a high % idle and a % iowait time greater than 25 percent, this might point to an I/O or disk-bound situation.

The following is an example extracted from an **iostat** command report:

```
...
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        223.5               0.2      4.2       70.0      25.5

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk3           2.7     163.2      20.4       1632         0
hdisk2           2.8     170.8      21.9       1708         0
hdisk0           0.0       0.0       0.0          0         0
hdisk1           2.1     175.6      17.3       1756         0
cd0              0.0       0.0       0.0          0         0
hdisk4          99.1     715.6     125.0          0      7156
```

The preceding example shows a high % iowait due to an I/O bottleneck on hdisk4.

Depending on the actual system, a high % iowait time could also be caused by excessive paging, due to a lack of real memory. It could also be due to unbalanced disk load, fragmented data, or usage patterns.

For an unbalanced disk load, the same **iostat** report provides the necessary information. But for information about file systems or logical volumes (which are logical resources) you have to use an AIX-specific tool, such as **filemon** or **fileplace**.

Alternatively, the **iostat** command can be used for determining that a performance problem is related to the CPU. Although **vmstat** should be the preferred tool for this analysis, in the absence of **vmstat** reports, **iostat** could be used. A good indication of a CPU bound problem is when % iowait time is 0 and the system is not idle (% idle = 0).

To investigate if a system does not have a memory problem, verify that the physical volume that is used for swapping does not have an excessive load. Use the **lsps -a** command to determine the physical volume of the swap area.

4.2.4 The iostat command on SMP systems

The calculation of I/O wait time on symmetrical multiprocessor (SMP) systems has been modified to provide a more accurate accounting of CPU utilization in commands such as **vmstat** and **iostat**.

Prior to AIX Version 4.3.3, the calculation of I/O wait time on SMP systems could result in inflated values (compared to uniprocessor (UP) systems). This was due to a statistical anomaly of the way AIX counted CPU time. The **vmstat** and **iostat** commands simply reported the CPU breakdown into the four categories of usr/sys/wio/idle as tabulated within the kernel. At each clock interrupt on each processor (100 times a second in AIX), a determination is made as to which of the four categories to place the last 10 ms of time in. If the CPU is busy in user mode at the time of the clock interrupt, usr gets the clock tick added into its category. If the CPU is busy in kernel mode at the time of the clock interrupt, the sys category gets the tick. If the CPU is not busy, a check is made to see if any disk I/O is in progress. If any disk I/O is in progress, the wio category is incremented. If no disk I/O is in progress and the CPU is not busy, then the idle category gets the tick. Notice in the prior discussion that it does not matter which processor starts the I/O. This fact leads to higher wio times on SMP systems compared to UP systems in some situations.

Since AIX Version 4.3.3, the I/O wait time is no longer inflated; all CPUs are no longer attributed wait time when a disk is busy and the CPU is idle. The decision is based on whether a thread is awaiting an I/O on the CPU being measured. This method can report accurate wio times when just a few threads are doing I/O and the system is otherwise idle.

4.2.5 Disk utilization report

Any potential disk I/O performance problem should be analyzed with the **iostat** command first. To only report the disk I/O, use the **iostat** command -d flag. In addition, the disk statistics can be limited to the selected disks by listing the physical volume names.

The **iostat** disk utilization report displays the following columns:

Disks Shows the names of the physical volumes. They are either disk or CD-ROM, followed by a number. By default, all drives are displayed unless the drives are specified in the command line.

% tm_act Indicates the percentage of time the physical disk was active. A drive is active during data transfer and command processing, such as seeking a new location. An increase in the disk active time percentage implies a performance decrease and response time increase. In general, when the utilization exceeds 40 percent, processes are waiting longer than necessary for I/O to complete, because most UNIX processes sleep while waiting for their I/O requests to complete.

Kbps Indicates the amount of data transferred (read or write) to the drive in KB per second. This is the sum of Kb_read plus Kb_wrtn, divided by the seconds in the reporting interval.

tps Indicates the number of transfers per second that were issued to the physical disk. A transfer is an I/O request at the device driver level to the physical disk. Multiple logical requests can be combined into a single I/O request to the disk. A transfer is of indeterminate size.

Kb_read Displays the total data (in KB) read from the physical volume during the measured interval.

Kb_wrtn Displays the amount of data (in KB) written to the physical volume during the measured interval.

When analyzing the drive utilization report and using the different data columns just described, it is important to notice the patterns and relationships between the data types.

There is normally a relationship between disk utilization %tm_act and data transfer rate tps. If the disk busy rate %tm_act is high, then the tps rate should also be high. However, if you get a high disk busy rate and a low disk transfer rate, you may have either a fragmented logical volume, file system, or individual file. Generally, you do need to be concerned about high disk busy when a disk is being used by a single AIX process (for example, a batch job). For example, if an application reads/writes sequentially, you should expect a high disk transfer rate (tps) and a high disk busy rate (%tm_act).

Kb_read and Kb_wrtn can confirm an understanding of an application's read/write behavior. However, they provide no information on the data access patterns.

An average physical volume utilization greater than 25 percent across all disks indicates an I/O bottleneck. The general conclusion of performance problems on disk, logical volume, and file system is that the more drives you have on your system, the better the disk I/O performance.

However, there is a limit to the amount of data that can be handled by the SCSI adapter; hence, the SCSI adapter could become a bottleneck. Especially on RS/6000 systems with SCSI-1 and SCSI-2 adapters, this could become an issue. To determine if a SCSI adapter is saturated, summarize all the KB/s values for disks located on the same adapter and compare the sum with the SCSI adapter throughput. In general, use 70 percent of the SCSI standard throughput rate. Examples of different SCSI types and their throughput is provided in the following:

- SCSI-1 throughput rate of 3.5 MB/s (70 percent of 5 MB/s)
- SCSI-2 throughput rate of 7 MB/s (70 percent of 10 MB/s)
- Ultra SCSI throughput rate of 28 MB/s (70 percent of 40 MB/s)

- Ultra2 SCSI throughput rate of 56 MB/s (70 percent of 80 MB/s)

If a saturated adapter is discovered, solve the problem by moving disks to other, less-used adapters already in the system or add an additional SCSI adapter.

For more information about improving disk I/O, see product documentation.

Note: As with the **vmstat** command, **iostat** can only give a first indication about a performance bottleneck. The system administrator will have to use more in-depth analysis tools such as **filemon** to identify the source of the slowdown.

4.3 The lockstat command

The **lockstat** command displays lock-contention statistics on SMP systems.

The AIX kernel locks generated on the systems can be verified and possible contentions identified.

Note: Before **lockstat** can be used, you must create, as root, a new boot image with the -L option to enable lock instrumentation:

```
# bosboot -a -d /dev/hdiskx -L
```

Where *x* is the number of the bootdisk.

The **lockstat** command generates a report for each kernel lock that meets all specified conditions. When no conditions are specified, the default values are used.

The syntax of the **lockstat** command is as follows:

```
lockstat [ -a ] [ -c LockCount ] [ -b BlockRatio ] [ -n CheckCount ]
[ -p LockRate ] [ -t MaxLocks ] [ interval [ count ] ]
```

The commonly used flags of the **lockstat** command are provided in Table 4-3.

Table 4-3 Commonly used flags of the lockstat command

Flag	Description
-c *LockCount*	Specifies how many times a lock must be requested during an interval in order to be displayed. A lock request is a lock operation, which in some cases cannot be satisfied immediately. All lock requests are counted. The default is 200.

Flag	Description
-b *BlockRatio*	Specifies a block ratio. When a lock request is not satisfied, it is said to be blocked. A lock must have a block ratio that is higher than BlockRatio to appear in the list. The default of BlockRatio is 5 percent.
-n *CheckCount*	Specifies the number of locks that are to be checked. The **lockstat** command sorts locks according to lock activity. This parameter determines how many of the most active locks will be subject to further checking. Limiting the number of locks that are checked maximizes system performance, particularly if **lockstat** is executed in intervals. The default value is 40.
-p *LockRate*	Specifies a percentage of the activity of the most-requested lock in the kernel. Only locks that are more active than this will be listed. The default value is 2, which means that the only locks listed are those requested at least 2 percent as often as the most active lock.
-t *MaxLocks*	Specifies the maximum number of locks to be displayed. The default is 10.

If the **lockstat** command is executed with no options, an output similar to the following would be displayed.

```
# lockstat
Subsys   Name                Ocn   Ref/s   %Ref   %Block   %Sleep
------------------------------------------------------------------
PFS      IRDWR_LOCK_CLASS   259   75356   37.49    9.44     0.21
PROC     PROC_INT_CLASS       1   12842    6.39   17.75     0.00
```

The **lockstat** command report contains the following data columns:

Subsys The subsystem to which the lock belongs.

Name The symbolic name of the lock class.

Ocn The occurrence number of the lock in its class.

Ref/s The reference rate, or number of lock requests per second.

%Ref The reference rate expressed as a percentage of all lock requests.

%Block The ratio of blocking lock requests to total lock requests. A block occurs whenever the lock cannot be taken immediately.

%Sleep The percentage of lock requests that causes the calling thread to sleep.

Some common subsystems are as follows:

PROC	Scheduler, dispatcher, or interrupt handlers
VMM	Pages, segment, and freelist
TCP	Sockets, NFS
PFS	inodes, icache

The name of the lock class defined in AIX can be found in the file /usr/include/sys/lockname.h. Some common classes are:

TOD_LOCK_CLASS	All interrupts that need the Time-of-Day (TOD) timer
PROC_INT_CLASS	Interrupts for processes
U_TIMER_CLASS	Per-process timer lock

The **lockstat** command can also be run in intervals similar to the **iostat** command, as shown in the following example:

```
# lockstat 10 100
```

The first number passed in the command line specifies the amount of time in seconds between each report. Each report contains statistics collected during the interval since the previous report. If no interval is specified, the system provides information covering an interval of one second and then exits. The second number determines the number of reports generated. It can only be specified if an interval is given.

Note: The **lockstat** command can be CPU intensive because there is overhead involved with lock instrumentation. That is the reason why it is not turned on by default. The overhead of enabling lock instrumentation is typically three to five percent. Be aware that AIX trace buffers will fill up much quicker when using this option because there are a lot of locks being used.

4.4 LVM performance analysis using the lslv command

There are various factors that affect logical volume (LV) performance; for example, the allocation position on the disk or the mirroring options. To obtain information about the logical volume, you can use the LVM **lslv** command, which provides information on:

LV attributes	List of the current logical volume settings
LV allocation	Placement map of the allocation of blocks on the disk
LV fragmentation	Fragmentation of the LV blocks

4.4.1 Logical volume attributes

Use the **lslv** command with no flags to view logical volume attributes, as shown in the following output:

```
# lslv mirrlv
LOGICAL VOLUME:     mirrlv                 VOLUME GROUP:   stripevg
LV IDENTIFIER:      000bc6fd1202118f.3     PERMISSION:     read/write
VG STATE:           active/complete        LV STATE:       closed/syncd
TYPE:               jfs                    WRITE VERIFY:   on
MAX LPs:            512                    PP SIZE:        16 megabyte(s)
COPIES:             2                      SCHED POLICY:   parallel
LPs:                120                    PPs:            240
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       maximum                RELOCATABLE:    yes
INTRA-POLICY:       inner middle           UPPER BOUND:    32
MOUNT POINT:        /u/mirrfs              LABEL:          None
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes
```

The previous example shows the LV attributes of a logical volume mirrlv, which is a mirrored logical volume located in the volume group stripevg.

For performance issues, the following attributes have to be taken into account:

- COPIES

  Indicates the number of physical copies. If copies equal 1, then the LV is un-mirrored. Values of 2 and 3 are used for mirrored LVs. The previous example has a copy value of 2.

- INTER-POLICY

  The inter-physical volume allocation policy specifies which policy should be used for choosing physical devices to allocate the physical partitions of a logical volume.

- INTRA-POLICY

  The intra-physical volume allocation policy specifies which strategy should be used for choosing physical partitions on a physical volume.

- MIRROR WRITE CONSISTENCY

  The mirror write consistency (MWC) ensures data consistency among mirrored copies of a logical volume during normal I/O processing. For every write to a logical volume, the LVM generates a write request to every mirrored copy. Mirror write consistency recovery should be performed for most mirrored logical volumes.

- WRITE VERIFY

  Specifies whether to verify all writes to the logical volume with a follow-up read. This option enhances availability, but decreases performance.

- SHED-POLICY

  Specifies one of the two types of scheduling policies used for logical volumes with multiple copies, sequential or parallel.

- BB POLICY

  Specifies whether to use Bad Block Relocation, which redirects I/O requests from a bad disk block to a valid one.

- RELOCATABLE

  Specifies whether to allow the relocation of the logical volume during volume group reorganization.

- UPPER BOUND

  Specifies the maximum number of physical volumes for allocation.

Mirroring

To enhance the availability of a logical volume, AIX supports data mirroring by providing multiple copies of the logical volumes on different disks.

When using mirroring, the write scheduling policies are:

- The parallel policy

  The parallel policy balances reads between the disks. On each read, the system checks whether the primary is busy. If it is not busy, the read is initiated on the primary. If the primary is busy, the system checks the secondary. If it is not busy, the read is initiated on the secondary. If the secondary is busy, the read is initiated on the copy with the least number of outstanding I/Os. Writes are initiated concurrently.

- The parallel/sequential policy

  The parallel/sequential policy always initiates reads on the primary copy. Writes are initiated concurrently.

- The parallel/round-robin policy

  The parallel/round robin policy is similar to the parallel policy except that instead of always checking the primary copy first, it alternates between the copies. This results in equal utilization for reads even when there is never more than one I/O outstanding at a time. Writes are initiated concurrently.

- The sequential policy

  The sequential policy results in all reads being issued to the primary copy. Writes happen serially, first to the primary disk; only when that is completed is the second write initiated to the secondary disk.

The parallel write scheduling policy provides the best performance and is the preferred option when creating the mirrored LV.

In general, the following recommendations provide the highest LVM availability:

- Use three logical partition copies (mirror twice) and include at least three physical volumes.
- Write verify should be switched on.
- Inter-disk policy should be set to minimum, which sets the mirroring copies equal to the number of physical volumes.
- Disk allocation policy should be set to strict, which establishes that no LP copies are on the same disk.
- The copies on the physical volumes should be attached to separate busses, adapters, and power supplies.

Providing the highest availability, however, can have a negative impact on LVM performance; therefore, some of the settings may need to be altered depending on your requirements.

Intra policy

The five LVM intra-allocation policies are: Inner edge, inner middle, center, outer middle, outer edge. The corresponding intra-disk positions' allocation policies are illustrated in Figure 4-5 on page 135.

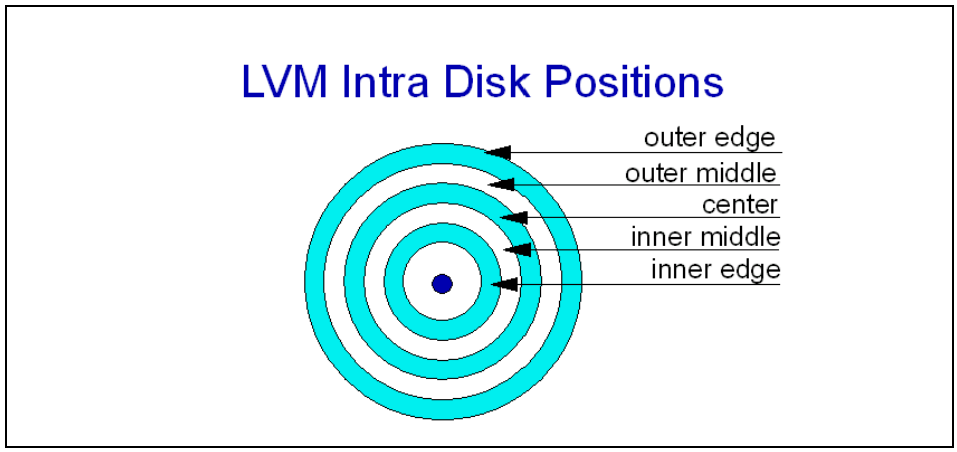


Figure 4-5 LVM intra-disk positions

In general, the performance of the intra-disk policies is as follows:

- The center allocation policy has the fastest average seek times on disks with < 4 GB of size. On larger disks, the outer edge has the fastest seek times.
- The outer middle and inner middle allocation policies provide reasonable average seek times. This is the default setting when creating a new logical volume.
- The outer edge (on disks < 4 GB) and inner edge policies have the slowest average seek times.

Inter policy

The possible inter-disk allocation policies are the MINIMUM and MAXIMUM. The MINIMUM inter-disk policy assigns the physical partitions to the logical volume on the same disk or to as few disks as possible. The MINIMUM policy provides the best availability.

The MAXIMUM inter-disk allocation policy allocates the physical partitions of the logical volume on as many disks as possible. The MAXIMUM policy provides the best performance.

For non-mirrored LVs, the MINIMUM policy indicates that one physical volume should contain all the physical partitions of this logical volume. If the allocation program must use two or more physical volumes, it uses the minimum number possible.

For mirrored LVs, the MINIMUM policy indicates that as many physical volumes as there are copies should be used. Otherwise, the minimum number of physical volumes possible is used to hold all the physical partitions.

Striping

Striping is a technique for spreading the data in a logical volume across several disk drives in such a way that the I/O capacity of the disk drives can be used in parallel to access data on the logical volume. The primary objective of striping is very high-performance reading and writing of large sequential files, but there are also benefits for random access. Figure 4-6 gives a simple example. When a striped LV is created, as many disks as possible should be used. Since AIX Version 4.3.3, mirroring of striped LVs is supported; therefore, the old conclusion that striping does not provide availability due to the lack of mirroring is no longer valid.

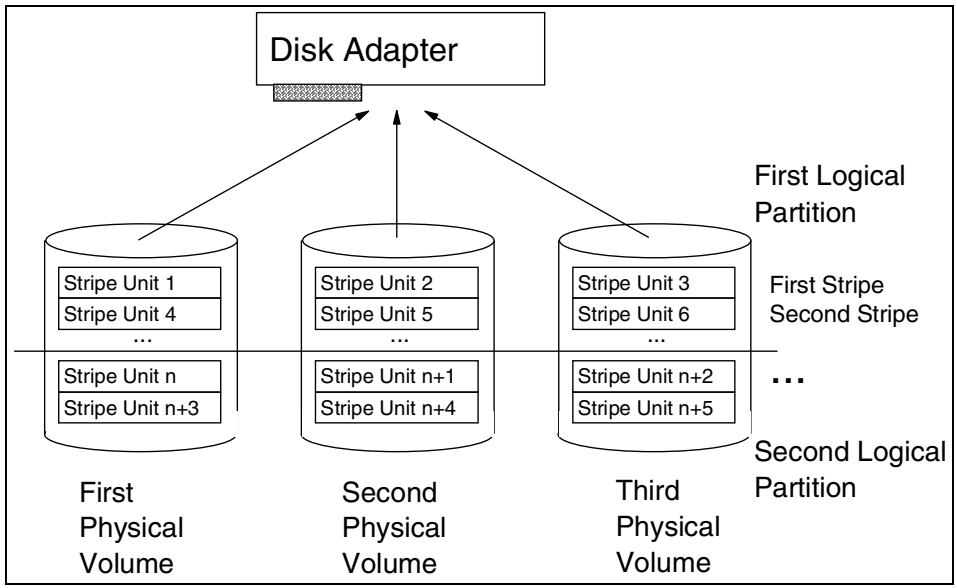


Figure 4-6 Striping a logical volume

When a striped logical volume is defined, then two additional LV attributes are displayed by the **lslv** command:

STRIPE WIDTH The number of stripes.

STRIPE SIZE The fixed size of each stripe block. Stripe size can be any power of 2 from 4 KB to 128 KB, but it is often set to 64 KB to get the highest levels of sequential I/O throughput.

Following is an example of a striped logical volume:

```
# lslv testlv
LOGICAL VOLUME:     testlv                 VOLUME GROUP:   rootvg
LV IDENTIFIER:      00015f8f00004c00000000efd92f4f69.13 PERMISSION:
read/write
VG STATE:           active/complete        LV STATE:       closed/syncd
```

```
TYPE:               jfs                     WRITE VERIFY:   off
MAX LPs:            512                     PP SIZE:        16 megabyte(s)
COPIES:             1                       SCHED POLICY:   striped
LPs:                32                      PPs:            32
STALE PPs:          0                       BB POLICY:      relocatable
INTER-POLICY:       maximum                 RELOCATABLE:    no
INTRA-POLICY:       middle                  UPPER BOUND:    2
MOUNT POINT:        N/A                     LABEL:          None
MIRROR WRITE CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes (superstrict)
STRIPE WIDTH:       2
STRIPE SIZE:        8K
```

RAID

Disk arrays are groups of disk drives that act like one disk as far as the operating system is concerned and which provide better availability or performance characteristics than the individual drives operating alone. Depending on the particular type of array that is used, it is possible to optimize availability or performance or to select a compromise between both. A summary of the RAID levels is provided in Table 4-4.

Table 4-4 RAID levels

RAID levels	Common name	Description	I/O request rate
0	Disk striping	Data distributed across the disks in the array. No redundant information provided.	Very high for both read and write
1	Mirroring	All data replicated on N separate disks. N is most commonly 2.	Up to N times that of a single disk for read, less than a single disk for write
3	Parallel transfer disks with parity	Each data sector is subdivided and distributed across all data disks. Redundant information is normally stored on a dedicated parity disk.	Similar to twice that of a single disk

RAID levels	Common name	Description	I/O request rate
5	RAID 5	Data sectors are distributed as with disk striping; redundant information is interspersed with user data.	Similar to disk striping for read; generally lower than single disk for write

4.4.2 Logical volume fragmentation

To check a logical volume for possible fragmentation, use the **lslv -l** command, as shown in the following example:

```
# lslv -l mirrlv
mirrlv:/u/mirrfs
PV               COPIES        IN BAND       DISTRIBUTION
hdisk2           120:000:000   90%           000:000:000:108:012
hdisk1           120:000:000   69%           000:000:000:083:037
```

This example uses the same LV mirrlv as in the last section.

The PV column indicates that the physical volume uses two disks (hdisk1 and hdisk2).

The COPIES column indicates that the total number of logical partitions (LP) is 120, and since it is a mirrored LV, both disks have the same amount of physical partitions (PPs) allocated (240 in total).

The IN BAND column indicates the level of intra-allocation policy as a percentage. If the LVM cannot meet the intra-policy requirement, it chooses the best alternative. In the above example, the intra-policy was *inner middle*, but only 69 percent on hdisk1 and 90 percent on hdisk2 could follow this allocation request.

The DISTRIBUTION column shows how the physical partitions are allocated in each section of the intra policy, as shown in the following relationship:

```
(outer edge) : (outer middle) : (center) : (inner middle) : (inner edge)
```

In this example, hdisk1 has allocated 83 PPs on the requested inner middle and 37 on the outer edge. Disk hdisk2 allocates the intra policy request better, hence the higher IN BAND level.

4.4.3 Logical volume allocation

To see the logical volume allocation of placement on the physical volume, use the following command:

```
# lslv -p hdisk1 mirrlv
hdisk1:mirrlv:/u/mirrfs
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE      1-10
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     11-20
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     21-30
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     31-40
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     41-50
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     51-60
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     61-70
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     71-80
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     81-90
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     91-100
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   USED          101-109

USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     110-119
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     120-129
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     130-139
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     140-149
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     150-159
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     160-169
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     170-179
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     180-189
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     190-199
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     200-209
USED   USED   USED   USED   USED   USED   USED   USED                   210-217

USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     218-227
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     228-237
USED   USED   USED   USED   USED   USED   FREE   FREE   FREE   FREE     238-247
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     248-257
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     258-267
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     268-277
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     278-287
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     288-297
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     298-307
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     308-317
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE                   318-325

USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     326-335
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     336-345
USED   USED   USED   USED   USED   0001   0002   0003   0004   0005     346-355
0006   0007   0008   0009   0010   0011   0012   0013   0014   0015     356-365
0016   0017   0018   0019   0020   0021   0022   0023   0024   0025     366-375
0026   0027   0028   0029   0030   0031   0032   0033   0034   0035     376-385
```

```
0036  0037  0038  0039  0040  0041  0042  0043  0044  0045     386-395
0046  0047  0048  0049  0050  0051  0052  0053  0054  0055     396-405
0056  0057  0058  0059  0060  0061  0062  0063  0064  0065     406-415
0066  0067  0068  0069  0070  0071  0072  0073  0074  0075     416-425
0076  0077  0078  0079  0080  0081  0082  0083                 426-433

0084  0085  0086  0087  0088  0089  0090  0091  0092  0093     434-443
0094  0095  0096  0097  0098  0099  0100  0101  0102  0103     444-453
0104  0105  0106  0107  0108  0109  0110  0111  0112  0113     454-463
0114  0115  0116  0117  0118  0119  0120  FREE  FREE  FREE     464-473
FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE     474-483
FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE     484-493
FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE     494-503
FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE     504-513
FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE     514-523
FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE     524-533
FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE  FREE         534-542
```

The output displays five sections that represent: Outer edge, outer middle, center, inner middle, and inner edge.

Each physical partition is marked with either a number or a keyword, which is described in the following:

Number A number indicates the logical partition number of the LV.

USED This keyword indicates that the physical partition at this location is used by another logical volume.

FREE This keyword indicates that this physical partition is not used by any logical volume. Logical volume fragmentation occurs if logical partitions are not contiguous across the disk.

STALE Although not present in the previous example, the STALE keyword indicates a physical partition that cannot be used.

This example shows that the one copy of mirrlv located on hdisk1 is allocated contiguously. The logical partitions (LPs) from 01–83 are allocated in the inner middle section, while the LPs 84–120 are allocated in the inner edge section.

When logical volumes are deleted, the physical partitions are freed, and this enables the space for either new logical volumes or the possibility of reorganizing the logical volumes, so that the LV fragmentation is limited. The LVM command **reorgvg** can reorganize logical volumes so that they comply with the intra-disk policies. By using **reorgvg** and providing both the volume group and the name of the logical volume, the highest priority is given to the listed volume group when performing the reorganization. During the reorganization, the volume group is locked and cannot be used.

4.4.4 Highest LVM performance

The following general recommendations can be used for creating logical volumes with high performance demands. However, keep in mind that when a logical volume is tuned for better performance, the availability may be impacted.

- No mirroring, which means the number of copies equals one (1).
- If mirroring is required then:
  - Write scheduling policy set to parallel.
  - Allocation policy set to strict, which means each copy is on separate physical volumes.
- Write verification set to no.
- Mirror write consistency (MWC) set to off.
- Intra policies:
  - Center: For *hot* logical volumes
  - Middle: For *moderate* logical volumes
  - Edge: For *cold* logical volumes
- Inter-disk allocation policy set to maximum, which mean that read/write operations are spread among physical volumes.

Additional performance improvement can be gained by creating a striped logical volume.

4.5 LVM and file system monitoring

To provide a more complete analysis of file system performance, AIX Version 4.3 provides a monitoring command, **`filemon`**. This provides information about a specific application or system I/O activity, assisting the problem determination process for performance tuning.

4.5.1 The filemon command

The **`filemon`** command monitors and presents trace data on the following four levels of file system utilization:

Logical file system The monitored operations include all read, write, open, and lseek system calls, which may or may not result in actual physical I/O, depending on whether the files are already buffered in memory. I/O statistics are kept on a per-file basis.

Virtual memory system	At this level, operations (that is, paging) between segments and their images on disk are monitored. I/O statistics are kept on a per-segment basis.
Logical volumes	I/O statistics are kept on a per-logical-volume basis.
Physical volumes	At this level, physical resource utilizations are obtained. I/O statistics are kept on a per-physical-volume basis.

Using the filemon command

The **filemon** command is based on the AIX trace facility to monitor I/O activity during a certain time interval. Because of this, **filemon** can be run only by root, and **filemon** cannot be executed in parallel with other trace-based commands, such as **tprof** and **netpmon**.

Tracing is started implicitly by the **filemon** command, but the trace can be controlled by the normal trace utilities: **trcstop**, **trcoff**, and **trcon**.

When tracing is stopped with **trcstop**, **filemon** writes a report either to stdout or to a specified file.

To specify the levels of data collected on all the layers, or on specific layers, use the -O layer option. The default is to collect data on the VM, LVM, and physical layers. Both summary and detailed reports are both generated.

Note: The **filemon** command will only collect data for those files opened after **filemon** was started, unless you specify the -u flag.

The following command sequence provides a simple example of **filemon** in action:

```
# filemon -o /tmp/filemonLF.out -O lf

Enter the "trcstop" command to complete filemon processing

# dd count=2048 if=/dev/zero of=/u/mirrfs/testMirrorFile
2048+0 records in.
2048+0 records out.
# dd count=2048 of=/dev/null if=/u/mirrfs/testMirrorFile
2048+0 records in.
2048+0 records out.
# trcstop
[filemon: Reporting started]
[filemon: Reporting completed]

[filemon: 10.666 secs in measured interval]
```

```
# ls -l filemonLF.out
-rw-r--r--   1 root     system      2627 Jul 07 12:51 filemonLF.out
#
```

The **filemon** command is started with the flag -O, specifying that only the logical file system (lf) data is to be monitored. This example uses two **dd** commands to write to a file and read from a file. The special devices /dev/zero and /dev/null are used to get clean read/write figures and make the reports more transparent. The output report of the **filemon** command in this example is placed in a dedicated file using the -o flag. The default is to write the report to the standard output.

4.5.2 Report analysis

The reports generated by **filemon** are dependent on the output level flag -O. The possible values for the output levels are:

lf	Logical file level
lv	Logical volume level
pv	Physical volume level
vm	Virtual memory level

The default value of -O is *all*. However, if -O is specified without a level, then lv, pv, and vm are the default.

The following section explains the **filemon** output reports using the examples from 4.5.1, “The filemon command” on page 141.

Logical file level report

The logical file level report, as shown in the following example, provides two sections, the Most Active Files Report, for information on the active files during the trace, and the Detailed File Stats Report, for detailed statistics on the individual files.

```
# cat /tmp/filemonLF.out
Fri Jul  7 12:51:38 2000
System: AIX server1 Node: 4 Machine: 000BC6FD4C00

Cpu utilization:  100.0%

Most Active Files
------------------------------------------------------------------------
  #MBs  #opns   #rds   #wrs  file                    volume:inode
------------------------------------------------------------------------
   2.0      2   2048   2048  testMirrorFile          /dev/mirrlv:17
   1.0      1   2048      0  zero
```

```
   1.0      1      0   2048  null
   0.0      3      6      0  ksh.cat                 /dev/hd2:23079
   0.0      2      2      0  dd.cat                  /dev/hd2:22970
   0.0      1      2      0  cmdtrace.cat            /dev/hd2:22947



------------------------------------------------------------------------
Detailed File Stats
------------------------------------------------------------------------

FILE: /u/mirrfs/testMirrorFile  volume: /dev/mirrlv  inode: 17
opens:                  2
total bytes xfrd:       2097152
reads:                  2048    (0 errs)
  read sizes (bytes):   avg   512.0 min     512 max     512 sdev     0.0
  read times (msec):    avg   0.003 min   0.000 max   0.084 sdev   0.005
writes:                 2048    (0 errs)
  write sizes (bytes):  avg   512.0 min     512 max     512 sdev     0.0
  write times (msec):   avg   0.028 min   0.012 max   0.443 sdev   0.044
lseeks:                 1
FILE: /dev/zero
opens:                  1
total bytes xfrd:       1048576
reads:                  2048    (0 errs)
  read sizes (bytes):   avg   512.0 min     512 max     512 sdev     0.0
  read times (msec):    avg   0.007 min   0.006 max   0.076 sdev   0.003

FILE: /dev/null
opens:                  1
total bytes xfrd:       1048576
writes:                 2048    (0 errs)
  write sizes (bytes):  avg   512.0 min     512 max     512 sdev     0.0
  write times (msec):   avg   0.001 min   0.000 max   0.023 sdev   0.002

FILE: /usr/lib/nls/msg/en_US/ksh.cat  volume: /dev/hd2 (/usr)  inode: 23079
opens:                  3
total bytes xfrd:       24576
reads:                  6       (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   0.033 min   0.000 max   0.085 sdev   0.036
lseeks:                 15

FILE: /usr/lib/nls/msg/en_US/dd.cat  volume: /dev/hd2 (/usr)  inode: 22970
opens:                  2
total bytes xfrd:       8192
reads:                  2       (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
```

```
  read times (msec):    avg   4.380 min   0.000 max   8.760 sdev   4.380
lseeks:                 10
FILE: /usr/lib/nls/msg/en_US/cmdtrace.cat  volume: /dev/hd2 (/usr)  inode:
22947
opens:                  1
total bytes xfrd:       8192
reads:                  2       (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   0.000 min   0.000 max   0.000 sdev   0.000
lseeks:                 8
```

The Most Active Files Report contains summary information of the most frequently used files during the monitoring period, defined in the following list.

#MBS Total number of megabytes transferred to and from the file. The rows are sorted by this field, in decreasing order.

#opns Number of times the file was opened during the measurement period.

#rds Number of read system calls made against the file.

#wrs Number of write system calls made against the file.

file Name of the file (full path name is in the detailed report).

volume:inode Name of volume that contains the file, and the file's inode number. This field can be used to associate a file with its corresponding persistent segment, shown in the virtual memory I/O reports. This field may be blank (for example, for temporary files created and deleted during execution).

The **filemon** example shows that the file testMirrorFile is the most active, with the 1 MB read and 1 MB write operations. Notice the read and write operations are made in 512 bytes units. This shows that the **dd** command used a 512-byte block size. The zero and null files do not have inodes because they are not connected to any file system, but are special device files.

From the **filemon** file-level report, it is very easy to see which files are generating the most I/O demand.

The Detailed File Stats Report provides information about every active file with the following details:

FILE Name of the file. The full path name is given, if possible.

volume Name of the logical volume/file system containing the file.

inode inode number for the file within its file system.

opens Number of times the file was opened while monitored.

total bytes xfrd Total number of bytes read/written to/from the file.

reads	Number of read calls against the file.
read sizes (bytes)	The read transfer-size statistics (avg, min, max, and sdev), in bytes.
read times (msec)	The read response-time statistics (avg, min, max, and sdev), in milliseconds.
writes	Number of write calls against the file.
write sizes (bytes)	The write transfer-size statistics.
write times (msec)	The write response-time statistics.
seeks	Number of lseek subroutine calls.

The detailed file level report from the previous **filemon** example is focusing on the file testMirrorFile. Here the read and write size of 512 bytes is even more evident. As it is the only read/write size used, the standard deviation (sdev) becomes 0. The read/write time is an interesting value in the detailed file statistics report. These can show, among other things, how the file system cache is performing.

Logical volume level report

The logical volume level report provides two sections: The Most Active Logical Volumes report and the Detailed Logical Volume Stats report.

The logical volume level report, provided by the following command, was generated from the same example in“Using the filemon command” on page 142, except the output level -O lv flags are used:

```
# filemon -o /tmp/filemonLF.out -O lv
```

The following is an excerpt from the logical volume level report:

```
...
Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume                  description
------------------------------------------------------------------------
  0.07      0   2016   64.5  /dev/mirrlv             /u/mirrfs
  0.00      0      8    0.3  /dev/loglv00            jfslog
  0.00      8      0    0.3  /dev/hd2                /usr
...
```

The headings are:

util	Utilization of the volume (fraction of time busy). The rows are sorted by this field, in decreasing order.
#rblk	Number of 512-byte blocks read from the volume.
#wblk	Number of 512-byte blocks written to the volume.

KB/sec Total transfer throughput in Kilobytes per second.

volume Name of volume.

description Contents of volume: Either a file system name or logical volume type (paging, jfslog, boot, or sysdump). Also indicates if the file system is fragmented or compressed.

This section of the logical volume level report shows clearly that the mirrlv is the most utilized LV. The report shows the transfer throughput of 64.5 KB/s for the mirrlv and its file system mirrfs. Notice also some activity on the loglv00, which is the jfslog for mirrfs.

Physical volume level report

The physical volume level report provides two sections: The Most Active Physical Volumes report and the Detailed Physical Volume Stats report.

The physical volume level report, provided by the following command, was generated with the same example as in “Using the filemon command” on page 142, except the output level -O pv was used:

```
# filemon -o /tmp/filemonLF.out -O pv
```

The following is an extraction of the physical volume level report:

```
...
Most Active Physical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume                   description
------------------------------------------------------------------------
  0.07      0   2096   66.4  /dev/hdisk1              N/A
  0.07      0   2080   65.9  /dev/hdisk2              N/A
  0.02      0    305    9.7  /dev/hdisk0              N/A
...
```

The headings are:

util Utilization of the volume (fraction of time busy). The rows are sorted by this field, in decreasing order.

#rblk Number of 512-byte blocks read from the volume.

#wblk Number of 512-byte blocks written to the volume.

KB/s Total volume throughput in Kilobytes per second.

volume Name of volume.

description Type of volume, for example, 9.1 GB disk or CD-ROM SCSI.

The physical volume level report of the **filemon** example shows almost equal activity of the two PVs, hdisk1 and hdisk2, because they are the mirrored copies of mirrlv.

Notice that hdisk1 has a slightly higher write block size than hdisk2 (and due to that, a slightly higher throughput). This is because the jfslog loglv00 is located on hdisk1.

Virtual memory level report

The virtual memory level report provides two sections: The Most Active Segments report and the Detailed VM Segment Stats report.

The virtual memory level report, provided by the following command, was generated with the same example as in"Using the filemon command" on page 142, except the output level -O vm was used:

```
# filemon -o /tmp/filemonLF.out -O vm
```

The following is an excerpt of a virtual memory level report:

```
...
Most Active Segments
------------------------------------------------------------------------
  #MBs  #rpgs  #wpgs  segid  segtype                  volume:inode
------------------------------------------------------------------------
   1.0      0    252   c473  page table
   0.0      0      1   fefe  log
...
```

The headings are:

#MBs Total number of megabytes transferred to and from the segment. The rows are sorted by this field, in decreasing order.

#rpgs Number of 4096-byte pages read into segment from disk (that is, page-in).

#wpgs Number of 4096-byte pages written from segment to disk (page-out).

segid Internal ID of segment.

segtype Type of segment: Working segment, persistent segment (local file), client segment (remote file), page table segment, system segment, or special persistent segments containing file system data (log, root directory, .inode, .inodemap, .inodex, .inodexmap, .indirect, .diskmap).

volume:inode For persistent segments, the name of the volume that contains the associated file, and the file's inode number. This field can be used to associate a persistent segment with its corresponding file, shown in the file I/O reports. This field is blank for non-persistent segments.

In this **filemon** example, the virtual memory level report does not contain any important information; it is merely mentioned for the completeness of the **filemon** reporting capabilities.

4.5.3 Typical AIX system behavior

When using the **filemon** command for performance analysis, the following issues should be kept in mind. The items listed are recommendations extracted from the documentation *RS/6000 Performance Tools in Focus*, SG24-4989.

Frequently accessed files:

- The /etc/inittab file is always very active. Daemons specified in /etc/inittab are checked regularly to determine whether they are required to be respawned.
- The /etc/passwd file is also very active, because file and directory access permissions is checked.

Disk access:

- A long seek time increases I/O response time and decreases performance.
- If the majority of the reads and writes require seeks, you may have fragmented files or overly active file systems on the same physical disk.
- If the number of reads and writes approaches the number of sequences, physical disk access is more random than sequential. Sequences are strings of pages that are read (paged in) or written (paged out) consecutively. The sequence length is the length, in pages, of the sequences. A random file access can also involve many seeks. In this case, you cannot distinguish from the **filemon** output if the file access is random or if the file is fragmented. If you have to further investigate with the **fileplace** command, see 4.9.2, “The fileplace command” on page 164, for more information.

Solutions to disk-bound problems:

- If large, I/O-intensive background jobs are interfering with interactive response time, you may want to activate I/O pacing.
- If it appears that a small number of files are being read over and over again, you should consider whether additional real memory would allow those files to be buffered more effectively.

- If the **iostat** command indicates that your workload I/O activity is not evenly distributed among the system disk drives, and the utilization of one or more disk drives is often 40-50 percent or more, consider reorganizing your file systems.
- If the workload's access pattern is predominantly random, you may want to consider adding disks and distributing the randomly accessed files across more drives.
- If the workload's access pattern is predominantly sequential and involves multiple disk drives, you may want to consider adding one or more disk adapters. It may also be appropriate to consider building a striped logical volume to accommodate large, performance-critical sequential files.

4.6 File system performance

These are some factors that affect file system performance.

- Dynamic allocation of resources may cause:
  - Logically contiguous files to be fragmented
  - Logically contiguous LVs to be fragmented
  - File blocks to be scattered
- Effects when files are accessed from disk:
  - Sequential access no longer sequential
  - Random access affected
  - Access time dominated by longer seek time

Once the file is in memory, these effects diminish.

Before going into analyzing the file system performance, an overview of how the AIX JFS is organized is in order.

4.6.1 AIX file system organization

In a journaled file system (JFS), files are stored in blocks of contiguous bytes. The default block size, also referred to as fragmentation size in AIX, is 4096 bytes (4 KB). The JFS inode contains an information structure of the file together with an array of eight pointers to data blocks. A file that is less than 32 KB is referenced directly from the inode.

A larger file uses a 4 KB block, referred to as an indirect block, for the addressing of up to 1024 data blocks. Using an indirect block, the file size of 4 MB (1024 x 4 KB) is possible.

For files larger than 4 MB, a second block, the double indirect block, is used. The double indirect block points to 512 indirect blocks, providing the possible addressing of 2 GB files (512 x 1024 x 4 KB). Figure 4-7 illustrates the addressing using double indirection.

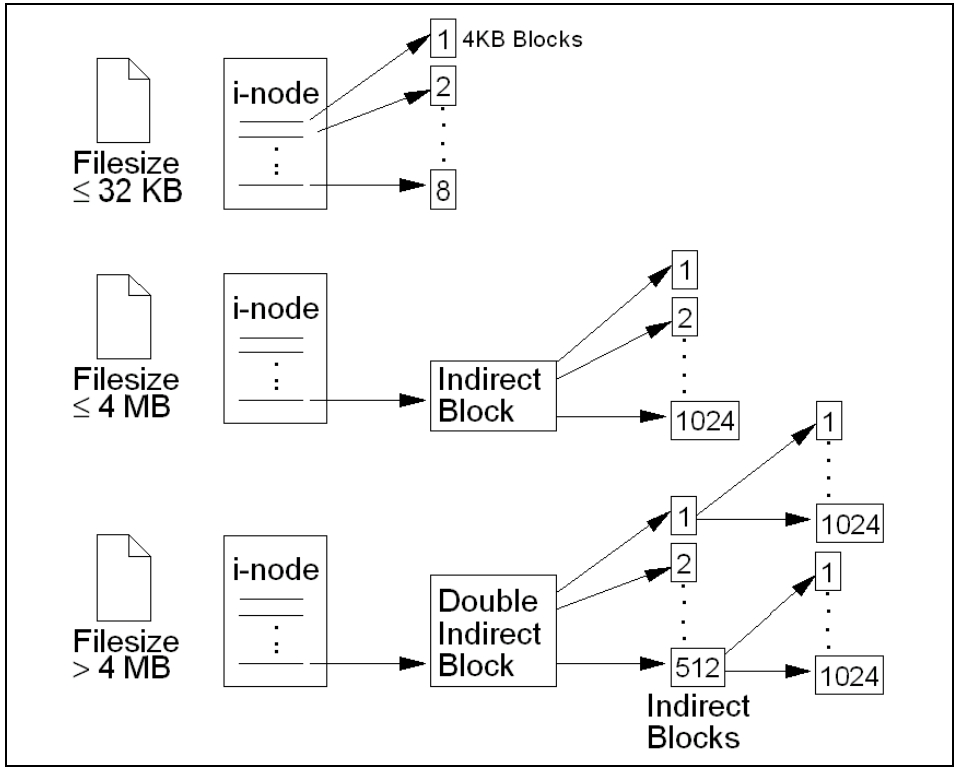


Figure 4-7 JFS organization

Since the introduction of AIX Version 4.2, support for even larger files has been added by defining a new type of JFS, the bigfile file system. In the bigfile file system, the double indirect are using references to 128 KB blocks rather than 4 KB blocks. However, the first indirect block still points to a 4 KB block, so that the large blocks are only used when the file size is above 4 MB. This provides a new maximum file size of just under 64 GB.

4.6.2 Enhanced journaled file system (JFS2)

Enhanced journaled file system supports the entire set of file system semantics. The file system uses database journaling techniques to maintain its structural consistency. This prevents damage to the file system when the file system is halted abnormally. Each JFS2 resides on a separate logical volume. The operating system mounts JFS2 during initialization. This multiple file system configuration is useful for system management functions such as backup, restore, and repair. It isolates a part of the file tree to allow system administrators to work on a particular part of the file tree.

The enhanced journaled file system (JFS2) supports multiple file system block sizes of 512, 1024, 2048, and 4096. Smaller block sizes minimize wasted disk space by more efficiently storing the data in a file or directory's partial logical blocks. Smaller block sizes also result in additional operational overhead.

The block size for a JFS2 is specified during its creation. Different file systems can have different block sizes, but only one block size can be used within a single file system.

- Variable number of inodes for enhanced journaled file system:

  JFS2 allocates inodes as needed. Therefore, the number of inodes available is limited by the size of the file system itself.

- Specifying file system block size:

  File system block size is specified during the file system's creation with the **crfs** and **mkfs** commands or by using the SMIT. The decision of file system block size should be based on the projected size of files contained by the file system.

- Identifying file system block size:

  The file system block size value can be identified through the **lsfs** command or the System Management Interface Tool (SMIT). For application programs, the statfs subroutine can be used to identify the file system block size.

- Compatibility and migration:

  The enhanced journaled file system (JFS2) is a new file system and is not compatible with any previous version of this operating system.

- Device driver limitations:

  A device driver must provide disk block addressability that is the same or smaller than the file system block size.

- Performance costs:

  Although file systems that use block sizes smaller than 4096 bytes as their allocation unit might require substantially less disk space than those using the default allocation unit of 4096 bytes, the use of smaller block sizes can incur performance degradation.

- Increased allocation activity:

  Because disk space is allocated in smaller units for a file system with a block size other than 4096 bytes, allocation activity can occur more often when files or directories are repeatedly extended in size. For example, a write operation that extends the size of a zero-length file by 512 bytes results in the allocation of one block to the file, assuming a block size of 512 bytes. If the file size is extended further by another write of 512 bytes, an additional block must be allocated to the file. Applying this example to a file system with 4096-byte blocks, disk space allocation occurs only once, as part of the first write operation. No additional allocation activity is performed as part of the second write operation since the initial 4096-byte block allocation is large enough to hold the data added by the second write operation.

- Increased block allocation map size:

  More virtual memory and file system disk space might be required to hold block allocation maps for file systems with a block size smaller than 4096 bytes. Blocks serve as the basic unit of disk space allocation, and the allocation state of each block within a file system is recorded in the file system block allocation map.

- Understanding enhanced journaled file system size limitations:

  The maximum size for an enhanced journaled file system is architecturally limited to 4 Petabytes. inodes are dynamically allocated by JFS2, so you do not need to consider how many inodes you may need when creating a JFS2 file system. You need to consider the size of the file system log.

- Enhanced journaled file system log size issues:

  In most instances, multiple journaled file systems use a common log configured to be 4 MB in size. When file systems exceed 2 GB or when the total amount of file system space using a single log exceeds 2 GB, the default log size might not be sufficient. In either case, scale log sizes upward as the file system size increases. The JFS log is limited to a maximum size of 256 MB.

- JFS2 file space allocation:

  File space allocation is the method by which data is apportioned physical storage space in the operating system. The kernel allocates disk space to a file or directory in the form of logical blocks. Logical block refers to the division of a file or directory contents into 512, 1024, 2048, or 4096 byte units. When a JFS2 file system is created the logical block size is specified to be one of 512, 1024, 2048, or 4096 bytes. Logical blocks are not tangible entities; however, the data in a logical block consumes physical storage space on the disk. Each file or directory consists of 0 or more logical blocks.

- Full and partial logical blocks:

  A file or directory may contain full or partial logical blocks. A full logical block contains 512, 1024, 2048, or 4096 bytes of data, depending on the file system block size specified when the JFS2 file system was created. Partial logical blocks occur when the last logical block of a file or directory contains less than the file system block size of data.

  For example, a JFS2 file system with a logical block size of 4096 with a file of 8192 bytes is two logical blocks. The first 4096 bytes reside in the first logical block and the following 4096 bytes reside in the second logical block. Likewise, a file of 4608 bytes consists of two logical blocks. However, the last logical block is a partial logical block containing the last 512 bytes of the file's data. Only the last logical block of a file can be a partial logical block.

- JFS2 file space allocation:

  The default block size is 4096 bytes. You can specify smaller block sizes with the **mkfs** command during a file system's creation. Allowable fragment sizes are 512, 1024, 2048, and 4096 bytes. You can use only one block's size in a file system.

  The kernel allocates disk space so that only the last file system block of data receives a partial block allocation. As the partial block grows beyond the limits of its current allocation, additional blocks are allocated.

  Block reallocation also occurs if data is added to logical blocks that represent file holes. A file hole is an empty logical block located prior to the last logical block that stores data. (File holes do not occur within directories.) These empty logical blocks are not allocated blocks. However, as data is added to file holes, allocation occurs. Each logical block that was not previously allocated disk space is allocated a file system block of space.

  Additional block allocation is not required if existing data in the middle of a file or directory is overwritten. The logical block containing the existing data has already been allocated file system blocks.

  JFS tries to maintain contiguous allocation of a file or directory's logical blocks on the disk. Maintaining contiguous allocation lessens seek time because the data for a file or directory can be accessed sequentially and found on the

same area of the disk. The disk space required for contiguous allocation may not be available if it has already been written to by another file or directory.

The file system uses a bitmap called the block allocation map to record the status of every block in the file system. When the file system needs to allocate a new block, it refers to the block allocation map to identify which blocks are available. A block can only be allocated to a single file or directory at a time.

- Extents:

  An extent is a sequence of contiguous file system blocks allocated to a JFS2 object as a unit. Large extents may span multiple allocation groups.

  Every JFS2 object is represented by an inode. inodes contain the expected object-specific information such as time stamps and file type (regular verses directory and others.) They also contain a B+ tree to record the allocation of extents.

  A file is allocated in sequences of extents. An extent is a contiguous variable-length sequence of file system blocks allocated as a unit. An extent may span multiple allocation groups. These extents are indexed in a B+ tree.

  There are two values needed to define an extent, the length and the address. The length is measured in units of the file system block size. 24-bit value represents the length of an extent, so an extent can range in size from 1 to 224 -1 file system blocks. Therefore the size of the maximum extent depends on the file system block size. The address is the address of the first block of the extent. The address is also in units of file system blocks. It is the block offset from the beginning of the file system.

  An extent-based file system combined with user-specified file system block size allows JFS2 to not have separate support for internal fragmentation. The user can configure the file system with a small file system block size (such as 512 bytes) to minimize internal fragmentation for file systems with large numbers of small-sized files.

  In general, the allocation policy for JFS2 tries to maximize contiguous allocation by allowing a minimum number of extents, with each extent as large and contiguous as possible. This allows for larger I/O transfer resulting in improved performance.

- Data fragmentation:

  JFS2 supports fragmented file systems. Fragmentation saves disk space by allowing a logical block to be stored on the disk in units or fragments smaller than the full block size of 4096 bytes. In a fragmented file system, only the last logical block of files no larger than 32 KB is stored in this manner, so that fragmented support is only beneficial for file systems containing numerous small files.

The use of fragments increases the potential for fragmentation of disk free space. Fragments allocated to a logical block must be contiguous on the disk. A file system experiencing free space fragmentation might have difficulty locating enough contiguous fragments for a logical block allocation, even though the total number of free fragments may exceed the logical block requirements. The JFS2 alleviates free space fragmentation by providing the defragfs program, which defragments a file system by increasing the amount of contiguous free space. The disk space savings gained from fragments can be substantial, while the problem of free space fragmentation remains manageable.

4.6.3 Journeled file system (JFS) log management

JFS logs enable rapid and clean recovery of file systems if a system goes down. If an application is doing synchronous I/O or is creating and removing many files in a short amount of time, there might be a lot of I/O going to the JFS log logical volume. If both the JFS log logical volume and the file system logical volume are on the same disk, this could cause an I/O bottleneck. We recommend migrating the JFS log device to another physical disk. It is also a good idea to create multiple JFS logs for a volume and assign them to specific file systems, preferably on fast-write cache devices, as these can provide for much better performance for log logical volumes (JFS logs or database logs).

You can have multiple log devices in a volume group. However, a log for a file system must be in the same volume group as that of the file system. A log logical volume or file system logical volume can be moved to another disk using the **migratepv** command, even while the system is running and in use. Remember to place the log logical volume on a physical volume different from your most active file system logical volume, as this will increase parallel resource usage.

Following is an example of how to create a new JFS log.

1. Create a new JFS log logical volume, as in the following example:

```
# mklv -t jfslog -y JFSlog testvg 1 hdisk2
JFSlog
```

   Or use the **smitty mklv** fastpath command shown in Figure 4-8 on page 157.

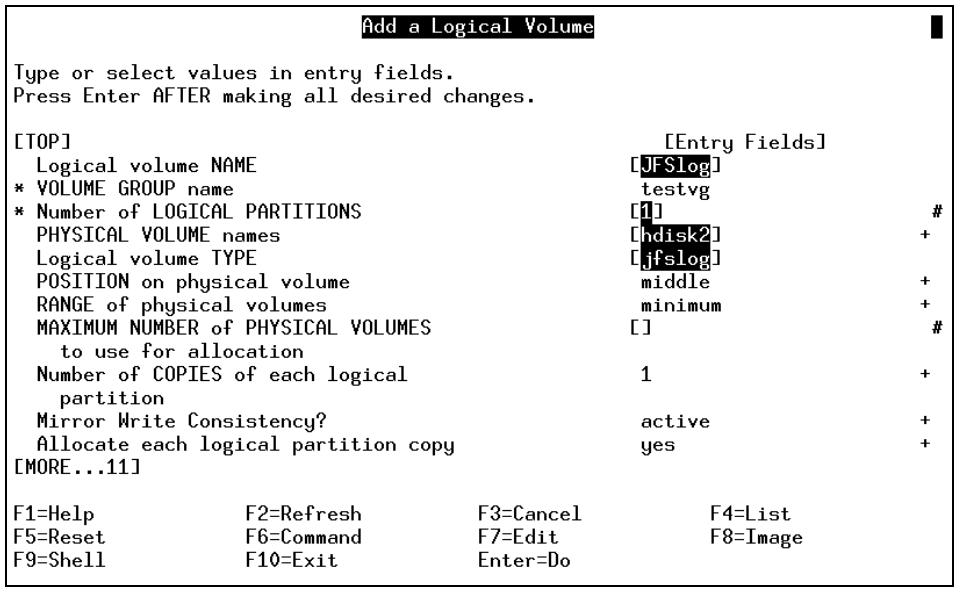


Figure 4-8 The smitty mklv add a logical volume dialog

2. Format the log, as in the following example:

```
# /usr/sbin/logform /dev/JFSlog
logform: destroy /dev/JFSlog (y)?y
```

3. Modify /etc/filesystems and the logical volume control block (LVCB), as in the following example:

```
# chfs -a log=/dev/JFSlog /testfilesystem
```

4. Unmount and then mount the file system. The mount option nointegrity may enhance local file system performance for certain write-intensive applications. This optimization eliminates writes to the JFS log. Note that the enhanced performance is achieved at the expense of metadata integrity. Therefore, use this option with extreme caution because a system crash can make a file system mounted with this option unrecoverable. The following is an example of the **mount** command:

```
# mount -o nointegrity /testfilesystem
```

4.6.4 The fileplace command

The use of files and file systems, depending on the application, can be very dynamic and can, over time, result in fragmentations that have impact on the file system performance, which influences the application performance.

Access to fragmented files may yield a large number of seeks and longer I/O response time. At some point, the system administrator may decide to reorganize the placement of files within the logical volume to reduce fragmentation and gain a more even distribution.

The **fileplace** command can assist in this task by displaying the placement of blocks in a file within a logical volume or within one or more physical volumes.

The **fileplace** command expects an argument containing the name of the file to examine, as shown in the following example:

```
# fileplace -iv sixMB

File: sixMB  Size: 6291456 bytes  Vol: /dev/restlv
Blk Size: 4096  Frag Size: 4096  Nfrags: 1536   Compress: no
Inode: 21  Mode: -rw-r--r--  Owner: root  Group: sys

DOUBLE INDIRECT BLOCK: 77000
INDIRECT BLOCKS: 75321 77001

  Logical Fragment
  ----------------
  0149576-0149583                   8 frags    32768 Bytes,   0.5%
  0075322-0075773                 452 frags  1851392 Bytes,  29.4%
  0149584-0150147                 564 frags  2310144 Bytes,  36.7%
  0077002-0077513                 512 frags  2097152 Bytes,  33.3%

  1536 frags over space of 74826 frags:  space efficiency = 2.1%
  4 fragments out of 1536 possible:  sequentiality = 99.8%
```

This example displays the logical fragmentation of a large file (6 MB). The general information displayed by **fileplace** is:

File	Name of the file.
Size	File size in bytes.
Vol	Name of the logical volume of the file system.
Blk Size	Physical block size 4 KB.
Frag Size	Fragment size; typically also 4 KB, but can be specified to values 512, 1 KB, or 2 KB at file system creation time.
Nfrags	The total amount of fragments used by the file.
Compress	Compression of file system. The default is no.
Inode	The inode reference number.
Mode/Owner/Group	General UNIX file system level inode information.

This file has, due to its size, both indirect blocks (75321, 77001) and a double indirect block (7700). This information is listed using the **fileplace -i** flag.

The first column under Logical Fragment shows the logical block numbers where the different parts of the file are. The next column shows the number of fragments that are contiguous and the amount of bytes in these contiguous fragments. The last number is the percentage of the block range compared to the total size.

Finally, the values for space efficiency and space sequentially are calculated when the **fileplace -v** flag is used. Higher space efficiency means files are less fragmented and will probably provide better sequential file access. Higher sequentially indicates that the files are more contiguously allocated, and this will probably be better for sequential file access.

Using **fileplace -p**, the physical block numbers and physical volume or volumes are shown.

The following is an example using **fileplace** on a mirrored logical volume:

```
# fileplace  -p /u/mirrfs/t5

File: /u/mirrfs/t5  Size: 504320 bytes  Vol: /dev/mirrlv
Blk Size: 4096  Frag Size: 4096  Nfrags: 124   Compress: no

  Physical Addresses (mirror copy 1)                                  Logical
Fragment
  ----------------------------------
----------------
  0320104-0320111  hdisk1            8 frags    32768 Bytes,   6.5%
0004168-0004175
  0319242-0319305  hdisk1           64 frags   262144 Bytes,  51.6%
0003306-0003369
  0319310-0319361  hdisk1           52 frags   212992 Bytes,  41.9%
0003374-0003425

  Physical Addresses (mirror copy 2)                                  Logical
Fragment
  ----------------------------------
----------------
  0320104-0320111  hdisk2            8 frags    32768 Bytes,   6.5%
0004168-0004175
  0319242-0319305  hdisk2           64 frags   262144 Bytes,  51.6%
0003306-0003369
  0319310-0319361  hdisk2           52 frags   212992 Bytes,  41.9%
0003374-0003425
```

This example shows the physical addresses used for the file t5 on the logical volume mirrlv. This file is physically located on both hdisk1 and hdisk2.

The **fileplace** command will not display NFS remote files. If a remote file is specified, the **fileplace** command returns an error message.

The **fileplace** command reads the file's list of blocks directly from the logical volume on disk. If the file is newly created, extended, or truncated, the information may not be on disk yet. Use the **sync** command to flush the information to the logical volume.

4.6.5 File system defragmentation

During the lifetime of a file system, a large number of files are created and deleted. This leaves, over time, a large number of gaps of free blocks. This fragmentation has a negative impact on the file system performance, as the newly created files become highly fragmented.

There is a simple way of organizing the free gaps in the file system. The **defragfs** command increases a file system's contiguous free space by reorganizing allocations to be contiguous rather than scattered across the disk. It does not increase the actual free space. The **defragfs** command is intended for fragmented and compressed file systems. However, you can also use the **defragfs** command to increase contiguous free space in non-fragmented file systems.

Another simple way of reorganizing the file system is to recreate the file system using a backup of the file system.

4.7 General recommendations for I/O performance

By using **lslv**, **fileplace**, **filemon**, and **iostat**, you can identify I/O, volume group, and logical volume problems. The following are general recommendations for how to achieve good LVM and file system performance and when to use the tools described in this chapter.

4.7.1 Logical volume organization for highest performance

The following general recommendations pertain to logical volume organization.

- Allocate hot LVs to different PVs to reduce disk contention.
- Spread hot LVs across multiple PVs so that parallel access is possible.
- Place the hottest LVs in the center of PVs, the moderate LVs in the middle of PVs, and the coldest LVs on edges of PVs so that the hottest logical volumes have the fastest access time.

- Mirroring can improve performance for read-intensive applications, but, as writes need to be performed several times, can impact the performance of other applications.

  If mirroring is needed, set the scheduling policy to parallel and the allocation policy to strict. Parallel scheduling policy will enable reading from the closest disk, and strict allocation policy allocates each copy on separate PVs.
- Make the LV contiguous to reduce access time.
- Set inter policy to maximum. This will spread each logical volume across as many physical volumes as possible, allowing reads and writes to be shared among several physical volumes.
- Place frequently used logical volumes close together to reduce the seek time.
- Set write verify to no so that there is no follow-up read (similar to a parity check) performed following a write.

4.7.2 Logical volume striping recommendations

The following are recommendations for logical volume striping.

- Spread the logical volume across as many physical volumes as possible.
- Use as many adapters as possible for the physical volumes.
- Create a separate volume group for striped logical volumes.
- Set a stripe-unit size of 64 KB.
- Set minpgahead to 2 (**vmtune** command).
- Set maxpgahead to 16 times the number of disk drives (**vmtune** command). This causes page-ahead to be done in units of the stripe-unit size (64 KB) times the number of disk drives, resulting in the reading of one stripe unit from each disk drive for each read-ahead operation.
- Request I/Os for 64 KB times the number of disk drives. This is equal to the maxpgahead value.
- Modify maxfree (**vmtune** command) to accommodate the change in maxpgahead (maxfree = minfree + maxpgahead).
- Use 64-byte aligned I/O buffers. If the logical volume will occupy physical drives that are connected to two or more disk adapters, the I/O buffers used should be allocated on 64-byte boundaries. This avoids having the LVM serialize the I/Os to the different disks. The following code would yield a 64-byte aligned buffer pointer:

```
char *buffer;
buffer = malloc(MAXBLKSIZE+64);
buffer = ((int)buffer + 64) & ~0x3f;
```

- If the striped logical volumes are on raw logical volumes and writes larger than 1.125 MB are being done to these striped raw logical volumes, increasing the lvm_bufcnt parameter with the **vmtune** command might increase throughput of the write activity.
- Also, it is not a good idea to mix striped and non-striped logical volumes in the same physical volume. All physical volumes should be the same size within a set of striped logical volumes.
- Limitations of striping:
  - Mirroring with striping prior to AIX Version 4.3.3 is not possible. On AIX 4.3.3 or later, the mirroring of logical volume striping is possible using the superstrict physical allocation policy.
  - Disk striping is mostly effective for sequential disk I/Os. With randomly accessed files, it is not as effective.

4.7.3 RAID recommendations

Select RAID 0 for applications that would benefit from the increased performance capabilities of this RAID level, but, because no data redundancy is provided, do not use RAID 0 for mission-critical applications that require high availability.

Select RAID 1 for applications where data availability is a key concern, and which have high levels of write operations, such as transaction files, and where cost is not a major concern.

Select RAID 3 for applications that process large blocks of data where write performance is not the key factor, and where the reduced cost of RAID 3 compared to mirroring is important.

Select RAID 5 for applications that manipulate small amounts of data, such as transaction processing applications. Since all the disk heads move independently to satisfy multiple requests, this is appropriate for multi-user applications. Also select RAID 5 as a good compromise between high performance, high availability, and low cost.

4.7.4 File system related performance issues

The following are file system performance issues.

- Create an additional log logical volume to separate the log of the most active file system from the default log. This will increase parallel resource usage.
- An `lslv` usage scenario:

  Determine if hot file systems are better located on a physical drive or spread across multiple physical drives.

- Some **filemon** usage scenarios:
  - Determine if hot files are local or remote.
  - Determine if paging space dominates disk utilization.
  - Look for heavy physical volume utilization. Determine if the type of drive (SCSI-1, SCSI-2, tape, and so on) or SCSI adapter is causing a bottleneck.
- Some **fileplace** usage scenarios:
  - Determine if the application performs a lot of synchronous (non-cached) file I/Os.
  - Look for file fragmentation. Determine if the hot files are heavily fragmented.

4.7.5 Paging space related disk performance issues

The following discusses paging space related disk performance issues.

- Never add more than one paging space on the same physical volume.
- Reorganize or add paging space to several physical volumes.

4.8 Overhead of using performance tools

As in any performance measurement on a system, each measurement consumes some resources. This is referred to as the *overhead* of the performance tools.

lslv This command mainly uses CPU time.

filemon This command can consume some CPU power. Use this tool with discretion, and analyze the system performance while taking into consideration the overhead involved in running the tool. In a CPU-saturated environment with a high disk-output rate, **filemon** slowed the writing program by about five percent.

fileplace Most variations of **fileplace** use fewer than 0.3 seconds of CPU time.

iostat The **iostat** command adds little overhead to the system. It uses about 20 milliseconds of CPU time for each report generated.

Note that the previous computing time measurements are from an RS/6000 Model 320.

4.9 Command summary

The following section provides a list of the key commands discussed in this chapter.

4.9.1 The filemon command

The **filemon** command monitors the performance of the file system, and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes. The command has the following syntax:

```
filemon [ -d ] [ -i file ] [ -o file] [ -O levels ] [ -P ] [ -T n] [ -u ] [-v ]
```

The commonly used flags are provided in Table 4-5.

Table 4-5 Commonly used flags of the filemon command

Flag	Description
-O *levels*	Monitors only the specified file system levels. Valid level identifiers are: lf (logical file level), vm (virtual memory level), lv (logical volume level), pv (physical volume level), and all (all is a short for lf, vm, lv, pv). If no -O flag is specified, the vm, lv, and pv levels are implied by default.
-o *file*	Name of the file where the output report is stored. If no flag is specified, the output is displayed on the standard output.
-u	Reports on files that were opened prior to the start of the trace daemon. The process ID (PID) and the file descriptor (FD) are substituted for the file name.

4.9.2 The fileplace command

The **fileplace** command displays the placement of file blocks within logical or physical volumes. The command has the following syntax and the flags are provided in Table 4-6:

```
fileplace [ { -l | -p } [ -i ] [ -v ] ] File
```

Table 4-6 Commonly used flags of the fileplace command

Flag	Description
-l	Displays file placement in terms of logical volume fragments for the logical volume containing the file. The -l and -p flags are mutually exclusive.

Flag	Description
-p	Displays file placement in terms of underlying physical volume, for the physical volumes that contain the file. If the logical volume containing the file is mirrored, the physical placement is displayed for each mirror copy. The -l and -p flags are mutually exclusive.
-i	Displays the indirect blocks for the file, if any. The indirect blocks are displayed in terms of either their logical or physical volume block addresses, depending on whether the -l or -p flag is specified.
-v	Displays more information about the file and its placement, including statistics on how widely the file is spread across the volume and the degree of fragmentation in the volume. The statistics are expressed in terms of either the logical or physical volume fragment numbers, depending on whether the -l or -p flag is specified.

4.9.3 The lslv command

The **lslv** command displays information about a logical volume. The command has the following syntax and the flags are provided in Table 4-7.

Display logical volume information:

```
lslv [ -L ] [ -l|-m ] [ -nPhysicalVolume ] LogicalVolume
```

Display logical volume allocation map:

```
lslv [ -L ] [ -nPhysicalVolume ] -pPhysicalVolume [ LogicalVolume ]
```

Table 4-7 Commonly used flags of the lslv command

Flag	Description
-l	Lists the fields for each physical volume in the logical volume.
-p *PhysicalVolume*	Displays the logical volume allocation map for the PhysicalVolume variable. If you use the LogicalVolume parameter, any partition allocated to that logical volume is listed by logical partition number.

4.10 Quiz

The following assessment questions help verify your understanding of the topics discussed in this chapter.

1. While a user is compiling a C program, **vmstat 120 10** is run to determine the cause of a performance problem. Given the **vmstat** output as shown in the exhibit indicates an I/O bottleneck, which of the following commands should be run next to get more information about the problem?

```
/usr/bin/vmstat 120 10
kthr   memory       page               faults             cpu
------ ------------ ------------------- ------------------
------------------
r  b   avm    fre    re pi  po  fr  sr  cy  in     sy     cs     us  sy  id     wa
0  1   59903  542    0  0   0   0   0   0   451    912    478    43  11  15     31
0  2   59904  550    0  0   0   0   0   0   521    1436   650    23  19  4      50
0  3   59950  538    0  0   0   0   0   0   344    649    249    7   7   6      80
0  2   59899  578    0  0   0   0   0   0   467    1829   500    12  14  4      70
0  2   59882  589    0  0   0   0   0   0   600    1292   705    6   8   3      61
0  3   59882  420    0  0   0   0   0   0   452    952    372    11  8   1      80
0  2   59954  420    0  0   0   0   0   0   537    1979   573    13  5   10     72
0  2   59954  423    0  0   0   0   0   0   618    1413   686    15  9   6      70
0  3   59954  420    0  0   0   0   0   0   551    938    634    4   2   2      92
0  2   59954  422    0  0   0   0   0   0   460    1376   496    14  2   4      80
```

   A. **lsps**

   B. **tprof**

   C. **iostat**

   D. **vmtune**

2. Which of the following commands should be used to show the percentage of time that the CPU(s) is idle waiting for pending system I/Os to complete?

   A. **tprof**

   B. **pstat**

   C. **iostat**

   D. **filemon**

3. Which of the following commands should be used to monitor disk utilization during the time a system is experiencing a disk I/O performance problem?

   A. **pstat**

   B. **iostat**

   C. **vmstat**

   D. **vmtune**

4. Which of the following utilities should be used to determine which disk or set of disks is experiencing contention on a SCSI bus?

   A. **lspv**

   B. **iostat**

   C. **lsdev**

   D. **vmstat**

5. Which of the following metrics provided by the **iostat** report is used to initially determine if a system is I/O bound?

   A. tin and tout

   B. % user and % sys

   C. % iowait and % tm_act

   D. Kb_read and Kb_wrtn

6. If the **filemon** command is invoked, which of the following indicates how to stop the command so that the **filemon** reports can be generated?

   A. Run **trcstop**.

   B. Run **filemon -u**.

   C. Perform a **kill -15** on the **filemon** PID.

   D. Perform a **kill -SIGSTOP** on the **filemon** PID.

7. Which of the following tools should be used to examine the details of a file's indirect inode, assuming it uses indirect inodes?

   A. **df**

   B. **istat**

   C. **filemon**

   D. **fileplace**

8. Which of the following tools will report how much of a logical volume is following its intra-policy request?

   A. **df -k**

   B. **lslv -l**

   C. **lsvg -l**

   D. **fileplace**

9. Which of the following is *true* of the **fileplace** command?

   A. A file can be placed in a specific location.

   B. The fragment distribution on a specified file is shown.

   C. Fragmentation is removed by rearranging file fragments.

   D. The distribution of all the files on a specified disk is shown.

10. Which of the following logical volume placement policies most likely provides the best performance for a logical volume that has Mirror Write Consistency Check turned on?

    A. INTRA-POLICY set to edge

    B. INTRA-POLICY set to center

    C. INTRA-POLICY set to middle

    D. INTRA-POLICY set to inner middle

11. There is a system where all rootvg logical volumes reside on the same disk (hdisk0), with the exception of the boot logical volume, which resides on another disk (hdisk1).

```
# filemon -O lv -o filemon.out;sleep 60;trcstop; cat filemon.out

Most Active Logical Volumes
------------------------------------------------------------------------
   util  #rblk     #wblk        KB/s   volume    description
------------------------------------------------------------------------
   0.84  105792    149280       177.1  /dev/hd9var  /var
   0.32  0         16800        11.9   /dev/hd8     jfslog
   0.01  4608      0            3.2    /dev/hd4     /
   0.02  55296     0            5.9    /dev/hd2     /usr
   0.01  2976      0            2.1    /dev/hd1     /home
```

    Using the **filemon** output as shown in the preceding exhibit, the workload across the disks should be balanced by:

    A. Moving the /var file system from hdisk0 to hdisk1

    B. Moving the hd5 logical volume from hdisk1 to hdisk0

    C. Creating a separate JFS log for the / (root) file system on hdisk1

    D. Creating a separate JFS log for the /var file system on hdisk1

12. If the percentage of RAM occupied by file pages falls below minperm, the page replacement algorithm steals:

    A. File pages only

    B. Computational only

    C. Both file and computational pages

    D. Persistent memory

13. An SMP system is running slowly and **vmstat** reflects high CPU idle time. Which of the following commands should be used to investigate contention?

    A. **vmtune**

    B. **schedtune**

    C. **filemon**

    D. **lockstat**

14. There is a mount option called nointegrity that bypasses the use of JFS logging. All of the following statements are true *except*:

    A. This option can provide better I/O performance, but can sacrifice a rapid and clean recovery of file systems.

    B. This option enables **fsck** to be done automatically at bootup of the file system should the system go down without a clean shutdown.

    C. This option can reduce I/O bottlenecks caused by a busy logical volume, especially if the system is creating and removing many files in a short amount of time.

    D. Fast-write cache devices can provide better performance for log logical volumes and can be a better option than the nointegrity.

15. An application is causing a response time issue. Which of the following can be used to determine the problem?

    A. **filemon**

    B. **fileplace**

    C. **rmss**

    D. **netstat**

4.10.1 Answers

The following are the preferred answers to the questions provided in this section.

1. C
2. C
3. B
4. B
5. C
6. A
7. D
8. B
9. B
10. A
11. A
12. C
13. D
14. B
15. A

4.11 Exercises

The following exercises provide sample topics for self study. They will help ensure comprehension of this chapter.

1. On a test system, with preferably two spare disks, create a testvg volume group. Create test logical volumes with the different parameters discussed in this chapter: Mirrored LV, intra-disk policy, inter-disk policy, strict policy, and striping.
2. Use the **lslv** command on the LVs created above and verify LV attributes, LV fragmentation, and LV allocation.
3. Perform a **filemon** trace on your test system using the following command sequence:

   ```
   # filemon -u -O lf,lv,pv -o /tmp/filemon.out ; sleep 30; tracestop
   ```

   Identify the most active files, logical volume, and disk drive.
4. On an existing file system, create a large file. Verify its fragmentation as well as space efficiency and sequence with the **fileplace** command.

5

Network performance tools

The following topics are discussed in this chapter:

- Network performance problems overview
- Network monitoring tools
- Network tuning tools

This chapter looks at network performance problems. It describes network examination and problem-solving procedures.

5.1 Overview

The recommended network performance tools to use first are the **ping** and the **netstat** commands. Usually, they provide you with enough information to discover your problems; if not, this chapter provides you with more insight into the network performance topic.

To understand the performance characteristics of the network subsystem in AIX, you must first understand some of the underlying architecture. Figure 5-1 on page 174 shows the path of data from an application on one system to another application on a remote system. The following discussion matches the diagram:

- As an application writes to a socket, the data is copied from the user space into the socket send buffer in the kernel space. Depending on the amount of data being copied into the socket send buffer, the socket puts the data into either mbufs or clusters. The size of the buffers in virtual memory that are used by the input is limited by the values:
  - udp_sendspace
  - tcp_sendspace
- Once the data is copied into the socket send buffer, the socket layer calls the transport layer (either TCP or UDP), passing it a pointer to the linked list of mbufs (an mbuf chain).
- If the size of the data is larger than the maximum transfer unit (MTU) of the LAN, one of the following is generally done:
  - TCP breaks the output into segments that comply with the MTU limit.
  - UDP leaves the breaking up of the output to the IP layer.
- If IP receives a packet larger than the MTU of the interface, it fragments the packet and sends the fragments to the receiving system, which reassembles them into the original packet.
- When the interface layer receives a packet from IP, it attaches the link-layer header information to the beginning of the packet and calls the device driver write routine.
- At the device-driver layer, the mbuf chain containing the packet is enqueued on the transmit queue. The maximum total number of output buffers that can be queued is controlled by the system parameter tx_que_size.
- Arriving packets are placed on the device driver's receive queue, and pass through the interface layer to IP. The maximum total number of input buffers that can be queued is controlled by the system parameter rx_que_size.

- If IP in the receiving system determines that IP in the sending system has fragmented a block of data, it coalesces the fragments into their original form and passes the data to TCP or UDP. If one of the fragments is lost in transmission, the incomplete packet is ultimately discarded by the receiver. The length of time IP waits for a missing fragment is controlled by the ipfragttl parameter.
  - TCP reassembles the original segments and places the input in the socket receive buffer.
  - UDP simply passes the input on to the socket receive buffer.

  The maximum size of IP's queue of packets received from the network interface is controlled by the ipqmaxlen parameter, which is set and displayed with the **no** command. If the size of the input queue reaches this number, subsequent packets are dropped.
- When the application makes a read request, the appropriate data is copied from the socket receive buffer in kernel memory into the buffer in the application's working segment.

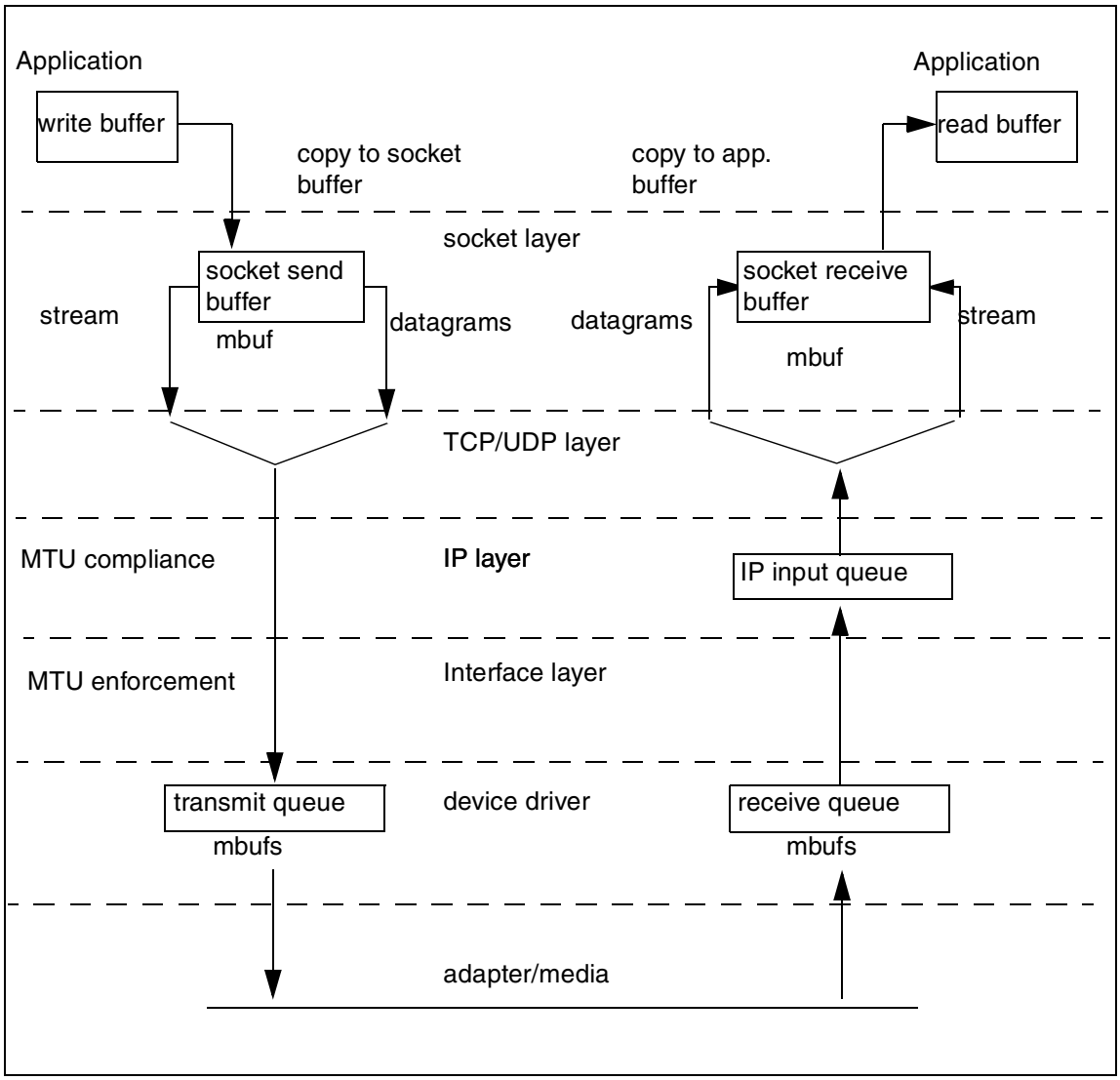


Figure 5-1 UDP/TCP/IP data flow

5.2 Adapter transmit and receive queue tuning

Most communication drivers provide a set of tunable parameters to control transmit and receive resources. These parameters typically control the transmit queue and receive queue limits, but may also control the number and size of

buffers or other resources. To check queue size for the ent0 adapter, use the **lsattr** command:

```
# lsattr -El ent0
busio         0x1000100      Bus I/O address                    False
busintr       15             Bus interrupt level                False
intr_priority 3              Interrupt priority                 False
tx_que_size   64             TRANSMIT queue size                True
rx_que_size   32             RECEIVE queue size                 True
full_duplex   no             Full duplex                        True
use_alt_addr  no             Enable ALTERNATE ETHERNET address True
alt_addr      0x000000000000 ALTERNATE ETHERNET address         True
```

To change queue size parameters, perform the following procedure.

Bring down the interface:

```
# ifconfig en0 detach
```

Change the value of the appropriate parameter:

```
# chdev -l ent0 -a tx_que_size=128
ent0 changed
```

Bring the interface back to the up state:

```
# ifconfig en0 up
```

To check if the queue's size should be changed, run the **netstat** command or adapter statistics utilities (**entstat**, **tokstat**, or others):

```
# netstat -v
ETHERNET STATISTICS (ent0) :
Device Type: IBM PCI Ethernet Adapter (22100020)
Hardware Address: 08:00:5a:fc:d2:e1
Elapsed Time: 0 days 0 hours 19 minutes 16 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 19                                   Packets: 0
Bytes: 1140                                   Bytes: 0
Interrupts: 0                                 Interrupts: 0
Transmit Errors: 0                            Receive Errors: 0
Packets Dropped: 0                            Packets Dropped: 0
                                              Bad Packets: 0

Max Packets on S/W Transmit Queue: 1
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 19                         Broadcast Packets: 0
Multicast Packets: 0                          Multicast Packets: 0
```

....

Two parameters should be checked:

- Max Packets on S/W Transmit Queue. This is the maximum number of outgoing packets ever queued to the software transmit queue. An indication of an inadequate queue size is if the maximal transmits queued equals the current queue size tx_que_size. This indicates that the queue was full at some point.
- S/W Transmit Queue Overflow. The number of outgoing packets that have overflowed the software transmit queue. A value other than zero indicates that the same actions needed if the Max Packets on S/W Transmit Queue reaches the tx_que_size should be taken. The transmit queue size has to be increased.

Adapter statistics utilities are quite useful. For example, the **entstat** command displays the statistics gathered by the specified Ethernet device driver. The user can optionally specify that the device-specific statistics be displayed in addition to the device-generic statistics. If no flags are specified, only the device-generic statistics are displayed. In the following example, to display the Ethernet device-generic statistics and the Ethernet device-specific statistics for ent0, enter:

```
# entstat -d ent0
-------------------------------------------------------------
ETHERNET STATISTICS (ent0) :
Device Type: IBM 10/100 Mbps Ethernet PCI Adapter (23100020)
Hardware Address: 00:06:29:b9:1f:08
Elapsed Time: 1 days 1 hours 24 minutes 55 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 443312                               Packets: 979962
Bytes: 198577053                              Bytes: 114525647
Interrupts: 2915                              Interrupts: 828638
Transmit Errors: 0                            Receive Errors: 0
Packets Dropped: 1                            Packets Dropped: 0
                                              Bad Packets: 0

Max Packets on S/W Transmit Queue: 24
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 1
```

5.3 Protocols tuning

The main goal in network performance tuning is to balance demands of users against resource constraints to ensure acceptable network performance.

You can do this with the following steps:

- Characterize workload, configuration, and bandwidth.
- Measure performance.
  - Run tools, identify bottlenecks.
  - Useful tools are **netstat**, **tcpdump**, and **iptrace**.
- Tune network parameters.
  - Make adjustments.
  - Useful tuning parameters are **no**, **nfso**, **chdev**, and **ifconfig**.

AIX allocates virtual memory for various TCP/IP networking tasks. The network subsystem uses a memory management facility called an mbuf. Mbufs are mostly used to store data for incoming and outbound network traffic. Having mbuf pools of the right size can have a very positive effect on network performance. Heavy network load can be a reason for low memory for the system, but too little virtual memory for network use can cause packet dropping. The dropped packet, on the other hand, can reduce the effective transmission throughput because of retransmissions or time outs.

The AIX operating system offers the capability for run-time mbuf pool configuration. There are a few system parameters that you can tune for this purpose:

thewall Kernel variable, controls the maximum amount of RAM (in kilobytes) that the mbuf management facility can allocate from the VMM.

tcp_sendspace Kernel variable, sets default socket send buffer. It keeps an application from overflowing the socket send buffer and limits the number of mbufs used by an application. The default value for tcp_sendspace is 16384.

tcp_recvspace Kernel variable, used as the default socket receive buffer size when an application opens a TCP socket. The default value for tcp_recvspace is 16384.

udp_sendspace Kernel variable, sets the limit for the amount of memory that can be used by a single UDP socket for buffering out-going data. If a UDP application fills this buffer space, it must sleep until some of the data has passed on to the next layer of the protocol stack. The default value for udp_sendspace is 9216.

udp_recvspace Kernel variable, sets the size limit of the receive space buffer for any single UDP socket. The default value for udp_recvspace is 41920.

rfc1323	If the value of this variable is non-zero, it allows the TCP window size to be the maximum of 32 bits instead of 16 bits. What this means is that you can set tcp_recvspace and tcp_sendspace to be greater than 64 KB.
sb_max	Kernel variable, controls the upper limit for any buffers.
ipqmaxlen	Kernel variable, controls length of the IP input queue. The default is 100 packets long, which is sufficient for single-network device systems. You may increase this value for systems with multiple network devices. The penalty for insufficient queue length is dropped packets.
ipforwarding	Specifies whether the kernel should forward packets. The default value of 0 prevents forwarding of IP packets when they are not for the local system. A value of 1 enables forwarding. ipforwarding is a runtime attribute.
udp_pmtu_discover	Enables or disables path MTU discovery for UDP applications. A value of 0 disables the feature, while a value of 1 enables it. This option is a runtime command.
tcp_pmtu_discover	Enables or disables path MTU discovery for TCP applications. A value of 0 disables the feature, while a value of 1 enables it. This option is a runtime command.

The operating system supports a path MTU discovery algorithm enabled by modifying udp_pmtu_discover and tcp_pmtu_discover with the **no** command. These options automatically force the size of all packets to not exceed the path MTU. When the path MTU has been discovered for a network route, a separate host route is cloned for the path. These cloned host routes, as well as the path MTU value for the route, can be displayed using the **netstat -r** command. Route expiration is controlled by the route_expire option of the **no** command.

Note: The values of the tcp or udp_sendspace and tcp or udp_recvspace variables must be less than or equal to the sb_max, so if you have reduced sb_max from its default, or want to use buffers larger than that default, you must also change the sb_max variable.

A network application that sends data in large bursts, such as a backup over the network, can generate socket buffer overflows. Insufficient buffer space for TCP sockets will merely limit throughput, but not inhibit proper operation. The TCP window limits the amount of data pending to be acknowledged and effectively limits the throughput of the sockets. The tcp_recvspace controls the TCP window size, which cannot be bigger than the socket buffer space. To increase performance of such an application, you have to remove the TCP window size

limit by setting the parameter rfc1323 to 1 and increasing the tcp_sendspace and tcp_recvspace values.

5.4 Network performance monitoring tools

This section describes the most common monitoring tools used to isolate network performance problems.

5.4.1 The vmstat command

You should invoke network monitoring tools in order to get more statistics for isolating a network bottleneck. When the **vmstat** command shows significant amounts of idle time that does not fit the problem, the system may be network bound. The following is a typical **vmstat** command report:

```
# vmstat 120 10
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr cy  in   sy  cs us sy id wa
 0  1  19331   824   0   0   0  0    0  0 636 1955 674  0  0 99  0
 0  1  19803   996   0   0   0  0    0  0 533 7466 591  0  0 99  0
 0  1  19974   860   0   0   0  0    0  0 822 4055 892  0  0 99  0
 0  1  19815   860   0   0   0  0    0  0 535 4096 509  0  0 99  0
 0  1  19816   855   0   0   0  0    0  0 577 4582 598  0  0 99  0
 0  1  19816   737   0   0   0  0    0  0 602 2720 672  0  0 99  0
 0  1  19895   724   0   0   0  0    0  0 616 3842 698  0  0 99  0
 0  1  17147   724   0   0   0  0    0  0 649 6427 626  0  0 99  0
 0  1  17065   720   0   0   0  0    0  0 516 3629 543  0  0 99  0
 0  1  17163   720   0   0   0  0    0  0 614 9030 688  0  0 99  0
 0  1  17343   720   0   0   0  0    0  0 420 8777 487  0  0 99  0
 0  1  17579   712   0   0   0  0    0  0 466 2182 473  0  0 99  0
 0  1  17647   712   0   0   0  0    0  0 497 3298 310  0  0 99  0
```

The disk I/O wait is in the wa column and the nondisk wait is in the id column. Nondisk wait includes network I/O wait or terminal I/O wait. If there is no terminal I/O wait, then the system is waiting for network I/O to complete. You should run one of the network monitoring tools to find out the reason for the network I/O wait.

The ping command

When you have connection problems, the first tool you should use is the **ping** command. It is used for investigating basic point-to-point network connectivity problems, answering questions about whether the remote host is attached to the network, and whether the network between the hosts is reliable. Additionally, **ping** can indicate whether a host name and IP address is consistent across

several machines. To check if the host server3 is *alive*, enter the following command:

```
# ping server3
PING server3: (9.3.240.58): 56 data bytes
64 bytes from 9.3.240.58: icmp_seq=0 ttl=255 time=1 ms
64 bytes from 9.3.240.58: icmp_seq=1 ttl=255 time=0 ms
^C
----server3 PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0/0/1 ms
```

This test checks round-trip times and packet loss statistics, as shown in the previous example.

5.4.2 The traceroute command

If you cannot reach a host that is in a different network, you can check the connection using the **traceroute** command. The **traceroute** output shows each gateway that the packet traverses on its way to find the target host, and the delay or network latency associated with that segment. If possible, examine the routing tables of the last machine shown in the **traceroute** output to check if a route exists to the destination from that host. The last machine shown is where the routing is going astray.

```
# traceroute 9.3.240.56
traceroute to 9.3.240.56 (9.3.240.56), 30 hops max, 40 byte packets
 1  server4e (10.47.1.1)  1 ms  1 ms  0 ms
 2  server1 (9.3.240.56)  1 ms  1 ms  1 ms
```

If the connections are performing poorly, packet fragmentation may be a problem. AIX Version 4.3 has a service that allows automatic path MTU discovery. A fixed MTU size can also be set with the **no** command.

5.4.3 The netstat command

The most common network monitoring tool is **netstat**. The **netstat** command is used to show the network status. It gives you an indication of the reliability of the local network interface. Traditionally, it is used more for problem determination than for performance measurement. It is useful in determining the amount of traffic on the network, therefore ascertaining whether performance problems are due to congestion.

There are various options to display:

- Active sockets
- Protocol statistics

- Device driver information
- Network data structures

To display statistics recorded by the memory management routines, use the **netstat** command with the -m flag. To enable more extensive statistics for network memory services (for AIX Version 4.3.2 and later), you should set kernel variable extendednetstats to 1 first, as shown in the following:

```
# no -o extendednetstats=1
# netstat -m
16 mbufs in use:
0 mbuf cluster pages in use
4 Kbytes allocated to mbufs
0 requests for mbufs denied
0 calls to protocol drain routines

Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed    free   hiwat   freed
32               97       102      0      31     640       0
64              124       805      0      68     320       0
128             111       923      0      17     160       0
256             152     41806      0      24     384       0
512              32       231      0      16      40       0
1024              1       158      0      19      20       2
2048              1       716      0       1      10       0
4096              2        14      0       7     120       0
8192              2       133      0       2      10       0
16384             1         1      0      12      24       7

By type       inuse     calls failed  memuse  memmax  mapb
mbuf             16     41218      0    4096   19712     0
mcluster          0       764      0       0    8192     0
socket          111       862      0   18048   18688     0
pcb              80       495      0   12480   12992     0
routetbl          8        15      0    1312    2080     0
ifaddr            7         7      0     832     832     0
mblk             66       435      0   15104   15488     0
mblkdata          2       294      0   16384   35840     0
strhead          11        48      0    3232    4256     0
strqueue         18       112      0    9216   11776     0
strmodsw         20        20      0    1280    1280     0
strosr            0        20      0       0     256     0
strsyncq         25       326      0    2688    3392     0
streams         137       245      0   14976   16256     0
devbuf            1         1      0     256     256     0
kernel table     14        15      0   45920   46432     0
```

```
temp              8        13      0    7424   15744     0

Streams mblk statistic failures:
0 high priority mblk failures
0 medium priority mblk failures
0 low priority mblk failures
```

The first paragraph of data shows how much memory is allocated to mbufs. The total number of bytes allocated for mbufs is the first statistic you should review. In this example, 4 KB is allocated out of a possible limit 16 MB. This limit can be regulated by the thewall kernel variable. The second statistic is named requests for mbufs denied. The nonzero value indicates that you should increase the limit by setting the thewall value. To check the thewall value, enter:

```
# no -o thewall
thewall = 16384
```

For network protocol statistics, use the **netstat** command with the -p flag and the appropriate protocol name. To receive statistics for IP protocol, use the command as follows:

```
# netstat -p IP
ip:
:
        59821 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        0 with bad options
        0 with incorrect version number
        7985 fragments received
        0 fragments dropped (dup or out of space)
        7 fragments dropped after timeout
        3989 packets reassembled ok
        55825 packets for this host
          ....
        47289 packets sent from this host
        8 packets sent with fabricated ip header
        0 output packets dropped due to no bufs, etc.
        0 output packets discarded due to no route
        11000 output datagrams fragmented
        22000 fragments created
        0 datagrams that can't be fragmented
          ....
        0 ipintrq overflows
```

When the ipintrq overflows counter has a nonzero value, you should change the length of the IP input queue using the **no** command:

```
# no -o ipqmaxlen=100
```

To display statistics for each protocol, enter:

```
# netstat -s -f inet|more
ip:
        191583 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        0 with bad options
        0 with incorrect version number
        0 fragments received
        0 fragments dropped (dup or out of space)
        0 fragments dropped after timeout
        0 packets reassembled ok
        164680 packets for this host
        37 packets for unknown/unsupported protocol
          ......
          ......
```

To check the amount of packets that pass through interfaces and the number of input/output errors, use the following command:

```
# netstat -i
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 link#1                          282515     0   283832     0     0
lo0   16896 127         localhost.austin.  282515     0   283832     0     0
lo0   16896 ::1                           282515     0   283832     0     0
en0   1500  link#2      8.0.5a.fc.d2.e1     49995     0    27187  3929     0
en0   1500  10.47       server4_            49995     0    27187  3929     0
tr0   1492  link#3      0.4.ac.61.73.f7    730283     0   317239   722     0
tr0   1492  9.3.240     server4f           730283     0   317239   722     0
```

5.4.4 The entstat command

The **entstat** command is a network analysis tool used for Ethernet networks. You should use this command to display the statistics gathered by the specified Ethernet device driver. You can also optionally specify that the device-specific statistics be displayed in addition to the device generic statistics. If no flags are specified, only the device-generic statistics are displayed. This command is also invoked when the **netstat** command is run with the -v flag. The **netstat** command does not issue any **entstat** command flags.

For example, to display the device generic statistics for ent0, enter:

```
# entstat ent0
-------------------------------------------------------------
ETHERNET STATISTICS (ent0) :
Device Type: IBM 10/100 Mbps Ethernet PCI Adapter (23100020)
Hardware Address: 00:06:29:b9:1f:08
Elapsed Time: 1 days 4 hours 54 minutes 56 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 16748                                Packets: 216281
Bytes: 6778818                                Bytes: 29241933
Interrupts: 48                                Interrupts: 211132
Transmit Errors: 25                           Receive Errors: 0
Packets Dropped: 1                            Packets Dropped: 0
                                              Bad Packets: 0
Max Packets on S/W Transmit Queue: 20
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 1

Broadcast Packets: 1160                       Broadcast Packets: 198718
Multicast Packets: 2                          Multicast Packets: 38
No Carrier Sense: 25                          CRC Errors: 0
DMA Underrun: 0                               DMA Overrun: 0
Lost CTS Errors: 0                            Alignment Errors: 0
Max Collision Errors: 0                       No Resource Errors: 0
Late Collision Errors: 0                      Receive Collision Erro
Deferred: 0                                   Packet Too Short Error
SQE Test: 1                                   Packet Too Long Errors
Timeout Errors: 0                             Packets Discarded by A
Single Collision Count: 0                     Receiver Start Count:
Multiple Collision Count: 0
Current HW Transmit Queue Length: 1

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Adapter Data Rate: 200
Driver Flags: Up Broadcast Running
        Simplex AlternateAddress 64BitSupport
        PrivateSegment DataRateSet
```

5.4.5 The fddistat command

The **fddistat** command is a network analysis tool used for FDDI networks. You should use this command to display the statistics gathered by the specified FDDI

device driver. If no flags are specified, only the device driver statistics are displayed. This command is also invoked when the **netstat** command is run with the -v flag. The **netstat** command does not issue any **fddistat** command flags.

If you issue an invalid Device_Name, the **fddistat** command will produce an error message stating that it could not connect to the device.

The -r flag resets all the statistics back to their initial values. This flag can only be issued by privileged users.

For example, to display the device driver statistics for fddi0, enter:

```
# fddistat fddi0
FDDI STATISTICS (fddi0) :
Elapsed Time: 0 days 0 hours 1 minutes 3 seconds
Transmit Statistics:
Receive Statistics:
Packets: 100                                Packets: 100
Bytes: 113800                               Bytes: 104700
Interrupts: 100                             Interrupts: 100
Transmit Errors: 0                          Receive Errors: 0
Packets Dropped: 0                          Packets Dropped: 0
Max Packets on S/W Transmit Queue: 0        Bad Packets: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0
Broadcast Packets: 0                        Broadcast Packets: 0
Multicast Packets: 0                        Multicast Packets: 0

General Statistics:

No mbuf Errors: 0
SMT Error Word: 00040080                    SMT Event Word: 000004a0
Connection Policy Violation: 0000           Port Event: 0000
Set Count Hi: 0000 Set Count Lo: 0003
Adapter Check Code: 0000                    Purged Frames: 0

ECM State Machine:        IN
PCM State Machine Port A: CONNECT
PCM State Machine Port B: ACTIVE
CFM State Machine Port A: ISOLATED
CFM State Machine Port B: CONCATENATED
CF State Machine:         C_WRAP_B
MAC CFM State Machine:    PRIMARY
RMT State Machine:        RING_OP

Driver Flags: Up Broadcast Running
        Simplex DualAttachStation
```

5.4.6 The tokstat

The **tokstat** command is a network analysis tool used for token ring networks. You should issue this command to display the statistics gathered by the specified token-ring device driver. You can optionally specify that the device-specific statistics be displayed in addition to the device-driver statistics. If no flags are specified, only the device-driver statistics are displayed. This command is also invoked when the **netstat** command is run with the -v flag. The **netstat** command does not issue any **tokstat** command flags.

For example, to display the device-driver statistics for tok0, enter:

```
# tokstat tok0
TOKEN-RING STATISTICS (tok0) :
Device Type: Token-Ring High-Performance Adapter (8fc8)
Hardware Address: 10:00:5a:4f:26:c1
Elapsed Time: 0 days 0 hours 8 minutes 33 seconds
Transmit Statistics:                  Receive Statistics:
--------------------                  -------------------
Packets: 191                          Packets: 8342
Bytes: 17081                          Bytes: 763227
Interrupts: 156                       Interrupts: 8159
Transmit Errors: 0                    Receive Errors: 0
Packets Dropped: 0                    Packets Dropped: 0
Max Packets on S/W Transmit Queue: 17 Bad Packets: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 1                 Broadcast Packets: 8023
Multicast Packets: 0                 Multicast Packets: 0
Timeout Errors: 0                    Receive Congestion Errors: 0
Current SW Transmit Queue Length: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0                    Lobe Wire Faults: 0
Abort Errors: 0                      AC Errors: 0
Burst Errors: 0                      Frame Copy Errors: 0
Frequency Errors: 0                  Hard Errors: 0
Internal Errors: 0                   Line Errors: 0
Lost Frame Errors: 0                 Only Station: 0
Token Errors: 0                      Remove Received: 0
Ring Recovered: 0                    Signal Loss Errors: 0
Soft Errors: 0                       Transmit Beacon Errors: 0
Driver Flags: Up Broadcast Running
   AlternateAddress ReceiveFunctionalAddr
```

5.4.7 The atmstat

The **atmstat** command shows Asynchronous Transfer Mode (ATM) Adapters statistics. You can optionally specify that the device-specific statistics be displayed in addition to the device-generic statistics. If no flags are specified, only the device-generic statistics are displayed.

To display the adapter-generic statistics for atm0, enter the following example:

```
# atmstat atm0
ATM STATISTICS (atm0) :
Device Type: Turboways 155 MCA ATM Adapter
Hardware Address: 08:00:5a:99:88:d5
Elapsed Time: 2 days 23 hours 38 minutes 18 seconds

Transmit Statistics:                        Receive Statistics:
--------------------                        -------------------
Packets: 50573                              Packets: 0
Bytes: 2225182                              Bytes: 0
Interrupts: 0                               Interrupts: 12904
Transmit Errors: 0                          Receive Errors: 0
Packets Dropped: 0                          Packets Dropped: 0
                                            Bad Packets: 0

Max Packets on S/W Transmit Queue: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Cells Transmitted: 50573                    Cells Received: 0
Out of Xmit Buffers: 0                      Out of Rcv Buffers: 0
Current HW Transmit Queue Length: 0         CRC Errors: 0
Current SW Transmit Queue Length: 0         Packets Too Long: 0
                                            Incomplete Packets: 0
                                              Cells Dropped: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Loss of Signals: 0
Adapter Reset Count: 0
Driver Flags: Up Running Simplex
   64BitSupport
Virtual Connections in use: 2
Max Virtual Connections in use: 2
Virtual Connections Overflow: 0
SVC UNI Version: auto_detect

Turboways ATM Adapter Specific Statistics:
------------------------------------------------
Packets Dropped - No small DMA buffer: 0
```

```
Packets Dropped - No medium DMA buffer: 0
Packets Dropped - No large DMA buffer: 0
Receive Aborted - No Adapter Receive Buffer: 0
Transmit Attempted - No small DMA buffer: 0
Transmit Attempted - No medium DMA buffer: 0
Transmit Attempted - No large DMA buffer: 0
Transmit Attempted - No MTB DMA buffer: 0
Transmit Attempted - No Adapter Transmit Buffer: 0
Max Hardware transmit queue length: 12
Small Mbuf in Use: 0
Medium Mbuf in Use: 0
Large Mbuf in Use: 64
Huge Mbuf in Use: 0
MTB Mbuf in Use: 0
Max Small Mbuf in Use: 0
Max Medium Mbuf in Use: 0
Max Large Mbuf in Use: 64
Max Huge Mbuf in Use: 0
MTB Mbuf in Use: 0
Small Mbuf overflow: 0
Medium Mbuf overflow: 0
Large Mbuf overflow: 0
Huge Mbuf overflow: 0
MTB Mbuf overflow: 0
```

5.4.8 The netpmon command

The **netpmon** command is the tool used for network I/O analysis. It uses **trace** as a means to collect statistics about events occurring in the network code in the kernel. Tracing must be stopped using a **trcstop** command. The **netpmon** command monitors a trace of system events, and reports on network activity and performance during the monitored interval. By default, the **netpmon** command runs in the background while one or more application programs or system commands are being executed and monitored. The **netpmon** command automatically starts and monitors a trace of network-related system events in real time. By default, the trace is started immediately; optionally, tracing may be deferred until the user issues a **trcon** command. When tracing is stopped by a **trcstop** command, the **netpmon** command generates all specified reports and exits. In the client-server environment, **netpmon** gives an excellent picture of how networking affects the overall performance. The **netpmon** command can be run on both the client and server. The **netpmon** command focuses on the following physical and logical resources:

CPU usage Monitors CPU usage by all threads and interrupt handlers. It estimates how much of this usage is due to network-related activities.

Network device-driver I/O Monitors I/O operations through network device drivers. In the case of transmission I/O, the command also monitors utilizations, queue lengths, and destination hosts.

Internet socket calls Monitors all subroutines on IP sockets.

NFS I/O Monitors read and write subroutines on client Network File System (NFS) files, client NFS remote procedure call (RPC) requests, and NFS server read or write requests.

The following example shows how network operation can impact the CPU performance. There was an NFS workload during this **netpmon** session.

```
# netpmon -O cpu; sleep 10 ; trcstop
on Jul 10 18:08:31 2000
System: AIX server1 Node: 4 Machine: 000BC6FD4C00

========================================================================

Process CPU Usage Statistics:
-----------------------------
                                                 Network
Process (top 20)             PID  CPU Time   CPU %   CPU %
-----------------------------------------------------------
kproc                          774    1.4956  24.896   0.000
kproc                          516    1.4940  24.870   0.000
kproc                         1032    1.4929  24.852   0.000
kproc                         1290    1.4854  24.727   0.000
kproc                         2064    0.0089   0.148   0.000
topas                        14798    0.0051   0.084   0.000
netpmon                      19204    0.0035   0.059   0.000
nfsd                         25054    0.0026   0.044   0.044
ksh                           5872    0.0010   0.016   0.000
dtterm                       17910    0.0009   0.014   0.000
netpmon                      22732    0.0007   0.012   0.000
trace                        28206    0.0006   0.010   0.000
swapper                          0    0.0005   0.008   0.000
xterm                        21984    0.0004   0.007   0.001
X                             4212    0.0003   0.005   0.000
trcstop                      11070    0.0002   0.004   0.000
java                         17448    0.0002   0.003   0.000
init                             1    0.0001   0.002   0.000
dtwm                         10160    0.0001   0.002   0.000
ot                           27694    0.0001   0.001   0.000
-----------------------------------------------------------
Total (all processes)                 5.9933  99.767   0.045
Idle time                             0.0000   0.000
```

```
========================================================================
/the output was edited for brevity/
```

Note that only the **nfsd** daemon consumed the network CPU time. The network CPU time means percentage of total time that the interrupt handler executed on behalf of network-related events. For other statistics, use the **netpmon** command with the -O flag and the appropriate keyword. The possible keywords are cpu, dd (network device driver I/O), so (Internet socket call I/O), nfs (NFS I/O), and all.

5.4.9 The tcpdump and iptrace commands

The tools discussed previously allow you to obtain a various number of statistics and network-type events in the AIX kernel. However, you might get a problem where statistic counters are not enough to find the cause of the problem. You may need to see the real data *crossing the wire*. There are two commands that let you see every incoming and outgoing packet from your interface: **tcpdump** and **iptrace**.

The **tcpdump** command prints out the headers of packets captured on a specified network interface. The following example shows a telnet session between hosts 9.3.240.59 and 9.3.240.58:

```
# tcpdump -i tr0 -n -I -t dst host 9.3.240.58
9.3.240.59.44183 > 9.3.240.58.23: S 1589597023:1589597023(0) win 16384 <mss
1452> [tos 0x10]
9.3.240.58.23 > 9.3.240.59.44183: S 1272672076:1272672076(0) ack 1589597024 win
15972 <mss 1452>
9.3.240.59.44183 > 9.3.240.58.23: . ack 1 win 15972 [tos 0x10]
9.3.240.59.44183 > 9.3.240.58.23: . ack 1 win 15972 [tos 0x10]
9.3.240.59.44183 > 9.3.240.58.23: P 1:16(15) ack 1 win 15972 [tos 0x10]
9.3.240.59.44183 > 9.3.240.58.23: P 1:16(15) ack 1 win 15972 [tos 0x10]
9.3.240.59.44183 > 9.3.240.58.23: . ack 6 win 15972 [tos 0x10]
9.3.240.59.44183 > 9.3.240.58.23: . ack 6 win 15972 [tos 0x10]
9.3.240.58.23 > 9.3.240.59.44183: P 6:27(21) ack 1 win 15972 (DF)
9.3.240.59.44183 > 9.3.240.58.23: P 1:27(26) ack 27 win 15972 [tos 0x10]
9.3.240.59.44183 > 9.3.240.58.23: P 1:27(26) ack 27 win 15972 [tos 0x10]
9.3.240.58.23 > 9.3.240.59.44183: P 27:81(54) ack 27 win 15972 (DF)
9.3.240.59.44183 > 9.3.240.58.23: P 27:30(3) ack 81 win 15972 [tos 0x10]
9.3.240.59.44183 > 9.3.240.58.23: P 27:30(3) ack 81 win 15972 [tos 0x10]
```

The first line indicates that TCP port 44183 on host 9.3.240.59 sent a packet to the telnet port (23) on host 9.3.240.58. The S indicates that the SYN flag was set. The packet sequence number was 1589597023 and it contained no data. There was no piggy-backed ack field; the available receive field win was 16384 bytes and there was a max-segment-size (mss) option requesting a mss of 1452 bytes. Host 9.3.240.58 replies with a similar packet, except it includes a piggy-backed ack field for host 9.3.240.59 SYN. Host 9.3.240.59 then acknowledges the host

9.3.240.58 SYN. The period (.) means no flags were set. The packet contains no data, so there is no data sequence number. On the eleventh line, host 9.3.240.59 sends host 9.3.240.58 26 bytes of data. The PUSH flag is set in the packet. On the twelfth line, host 9.3.240.58 says it received data sent by host 9.3.240.59 and sends 54 bytes of data; it also includes a piggy-backed ack for sequence number 27.

The **iptrace** daemon records IP packets received from configured interfaces. Command flags provide a filter so that the daemon traces only packets meeting specific criteria. Packets are traced only between the local host on which the **iptrace** daemon is invoked and the remote host. To format **iptrace** output, run the **ipreport** command. The following example shows the query from host 9.3.240.59 to DNS server 9.3.240.2. The output from the **nslookup** command is shown in the following:

```
# nslookup www.prokom.pl
Server:  dhcp240.itsc.austin.ibm.com
Address:  9.3.240.2

Non-authoritative answer:
Name:    mirror.prokom.pl
Address:  153.19.177.201
Aliases:  www.prokom.pl
```

The data was captured by the **iptrace** command, similar to the following:

```
# iptrace -a -P UDP -s 9.3.240.59 -b -d 9.3.240.2 /tmp/dns.query
```

The output form the **iptrace** command was formatted by the **ipreport** command, as follows:

```
TOK: ====( 81 bytes transmitted on interface tr0 )==== 17:14:26.406601066
TOK: 802.5 packet
TOK: 802.5 MAC header:
TOK: access control field = 0, frame control field = 40
TOK: [ src = 00:04:ac:61:73:f7, dst = 00:20:35:29:0b:6d]
TOK: 802.2 LLC header:
TOK: dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP:  < SRC =      9.3.240.59 >  (server4f.itsc.austin.ibm.com)
IP:  < DST =       9.3.240.2 >  (dhcp240.itsc.austin.ibm.com)
IP:  ip_v=4, ip_hl=20, ip_tos=0, ip_len=59, ip_id=64417, ip_off=0
IP:  ip_ttl=30, ip_sum=aecc, ip_p = 17 (UDP)
UDP: <source port=49572, <destination port=53(domain) >
UDP: [ udp length = 39 | udp checksum = 688d ]
DNS Packet breakdown:
    QUESTIONS:
   www.prokom.pl, type = A, class = IN

TOK: ====( 246 bytes received on interface tr0 )==== 17:14:26.407798799
```

```
TOK: 802.5 packet
TOK: 802.5 MAC header:
TOK: access control field = 18, frame control field = 40
TOK: [ src = 80:20:35:29:0b:6d, dst = 00:04:ac:61:73:f7]
TOK: routing control field = 02c0,  0 routing segments
TOK: 802.2 LLC header:
TOK: dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP:  < SRC =       9.3.240.2 >  (dhcp240.itsc.austin.ibm.com)
IP:  < DST =      9.3.240.59 >  (server4f.itsc.austin.ibm.com)
IP:  ip_v=4, ip_hl=20, ip_tos=0, ip_len=222, ip_id=2824, ip_off=0
IP:  ip_ttl=64, ip_sum=7cc3, ip_p = 17 (UDP)
UDP: <source port=53(domain), <destination port=49572 >
UDP: [ udp length = 202 | udp checksum = a7bf ]
DNS Packet breakdown:
    QUESTIONS:
   www.prokom.pl, type = A, class = IN
    ANSWERS:
    ->  www.prokom.plcanonical name = mirror.prokom.pl
    ->  mirror.prokom.plinternet address = 153.19.177.201
    AUTHORITY RECORDS:
    ->  prokom.plnameserver = phobos.prokom.pl
    ->  prokom.plnameserver = alfa.nask.gda.pl
    ->  prokom.plnameserver = amber.prokom.pl
    ADDITIONAL RECORDS:
    ->  phobos.prokom.plinternet address = 195.164.165.56
    ->  alfa.nask.gda.plinternet address = 193.59.200.187
    ->  amber.prokom.plinternet address = 153.19.177.200
```

There are two packets shown on the **ipreport** output above (the key data is shown in bold face text). Every packet is divided into a few parts. Each part describes a different network protocol levels. There are the token ring (TOK), IP, UDP, and the application (DNS) parts. The first packet is sent by host 9.3.240.59 and is a query about the IP address of the www.prokom.pl host. The second one is the answer.

5.5 Network performance management tools

Use the **no** command to configure network attributes. The **no** command sets or displays current network attributes in the kernel. This command only operates on the currently running kernel. The command must be run again after each startup or after the network has been configured. To make changes permanent, make changes to the appropriate /etc/rc. file. To list the current value of every parameter you can change, use the following command:

```
# no -a
         extendednetstats = 1
                  thewall = 18420
```

```
        sockthresh = 85
            sb_max = 1048576
         somaxconn = 1024
                .....
         lowthresh = 90
         medthresh = 95
          psecache = 1
    subnetsarelocal = 1
             maxttl = 255
          ipfragttl = 60
    ipsendredirects = 1
       ipforwarding = 1
            udp_ttl = 30
            tcp_ttl = 60
         arpt_killc = 20
      tcp_sendspace = 16384
      tcp_recvspace = 16384
      udp_sendspace = 9216
      udp_recvspace = 41920
                .....
```

To change the value of the thewall system parameter, shown as 18420, to 36840, use the **no** command, as shown in the following.

```
# no -o thewall=36840
```

The **ifconfig** command can be used to assign an address to a network interface or to configure or display the current configuration information. For tuning purposes, it is used to change the MTU size:

```
# ifconfig en0 mtu 1024
```

Note: The MTU parameters have to be the same on all nodes of the network.

The **chdev** command is also used to change the value of system attributes. The changes made by the **chdev** command are permanent because they are stored in the ODM database. To display the current value of the parameters of the en0 interface, use the **lsattr** command, as follows.

```
# lsattr -El en0
mtu           1500        Maximum IP Packet Size for This Device     True
remmtu        576         Maximum IP Packet Size for REMOTE Networks True
netaddr       10.47.1.6   Internet Address                           True
state         up          Current Interface Status                   True
arp           on          Address Resolution Protocol (ARP)          True
netmask       255.255.0.0 Subnet Mask                                True
security      none        Security Level                             True
authority                 Authorized Users                           True
broadcast                 Broadcast Address                          True
```

```
netaddr6                N/A                                     True
alias6                  N/A                                     True
prefixlen               N/A                                     True
alias4                  N/A                                     True
rfc1323                 N/A                                     True
tcp_nodelay             N/A                                     True
tcp_sendspace           N/A                                     True
tcp_recvspace           N/A                                     True
tcp_mssdflt             N/A                                     True
```

To permanently change value of the MTU parameter, enter:

```
# chdev -l en0 -a mtu=1024
en0 changed
```

5.6 Name resolution

If a network connection seems inexplicably slow at times but reasonable at other times, you should check the name resolution configuration for your system. To perform a basic diagnostic for name resolving, you can use either the **host** command or the **nslookup** command.

```
# host dhcp240.itsc.austin.ibm.com
dhcp240.itsc.austin.ibm.com is 9.3.240.2
```

The name resolution can be served through either the remote DNS server or the remote NIS server. If one of them is down, you have to wait until a TCP time-out occurs. The name can be resolved by an alternate source, which can be a secondary name server or the local /etc/hosts file.

First check the /etc/netsvc.conf file or NSORDER environment variable for your particular name resolution ordering. Then check the /etc/resolv.conf file for the IP address of the name server and try to **ping** it. If you can **ping** it, then it is up and reachable. If not, try a different name resolution ordering.

Resolver routines on hosts running TCP/IP attempt to resolve names using sources listed in Table 5-1.

Table 5-1 Default search order for the nslookup command

Default nsorder lookup	Description
1. BIND / DNS	Resolver routines will look for /etc/resolv.conf by default.
2. NIS	NIS is queried if it is running. NIS is authoritative over the local /etc/hosts.

Default nsorder lookup	Description
3. /etc/hosts	If NIS is not running then the local /etc/host file is searched.

5.7 NFS performance tuning

This section discusses the NFS server and the NFS client performance issue.

5.7.1 NFS server-side performance

When narrowing down the performance discussion on servers to the NFS specifics, the issue is often related to dropped packages. NFS servers may drop packets due to overloads.

One common place where a server will drop packets is the UDP socket buffer. The default for AIX Version 4.3 is TCP for data transfer, but UDP is still used for mounting. Dropped packets here are counted by the UDP layer, and the statistics can be seen by using the **netstat -p UDP** command. For example:

```
# netstat -p UDP
udp:
        89827 datagrams received
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
        329 dropped due to no socket
        77515 broadcast/multicast datagrams dropped due to no socket
        0 socket buffer overflows
        11983 delivered
        11663 datagrams output
(At the test system the buffer size was sufficient)
```

NFS packets will usually be dropped at the socket buffer only when a server has a lot of NFS write traffic. The NFS server uses a UDP and TCP socket attached to the NFS port, and all incoming data is buffered on those ports. The default size of this buffer is 60000 bytes. Dividing that number by the size of the default NFS Version 3 write packet (32765), you find that it will take only two simultaneous write packets to overflow that buffer. That could be done by just one NFS client (with the default configurations). It is not as easy as it sounds to overflow the buffer in normal system operation. As soon as the first packet reaches the socket, an nfsd will be awakened to start taking the data off.

One of two things has to happen for packets to be dropped. There must be high volume or a high burst of traffic on the socket. If there is high volume, a mixture of many writes plus other possibly non-write NFS traffic, there may not be enough nfsd daemons to take the data off the socket fast enough to keep up with the volume (it takes a dedicated nfsd to service each NFS call of any type). In the high burst case, there may be enough nfsds, but the speed at which packets arrive on the socket is such that the nfsd daemons cannot wake up fast enough to keep it from overflowing.

Each of the two situations has a different resolution. In the case of high volume, it may be sufficient to just increase the number of nfsd daemons running on the system. Since there is no significant penalty for running with more nfsd daemons on an AIX machine, this should be tried first. This can be done with the following command:

```
# chnfs -n 16
```

This will stop the currently running daemons, modify the SRC database code to reflect the new number, and restart the daemons indicated.

In the case of a high burst of traffic, the only solution is to make the socket bigger in the hope that some reasonable size will be sufficiently large enough to give the nfsd daemons time to catch up with the burst. Memory dedicated to this socket will not be available for any other use so it must be understood that making the socket larger may result in memory that will be under utilized the vast majority of the time. The cautious administrator will watch the socket buffer overflows statistic and correlate it with performance problems and make a determination on how big to make the socket buffer.

The **nfso** command is a great utility to list and configure NFS network variables. The syntax for this command is:

```
nfso { -a | -d Option | -l HostName | -o Option [ =NewValue ] } [ -c ]
```

Note: The **nfso** command performs no range checking; therefore, it accepts all values for the variables. If used incorrectly, the **nfso** command can make your system inoperable.

For example, to check the NFS kernel options of nfs_socketsize, nfs_tcp_socketsize, use the **nfso** command:

```
# nfso -a
portcheck= 0
udpchecksum= 2
nfs_socketsize= 60000
nfs_tcp_socketsize= 60000
nfs_setattr_error= 0
nfs_gather_threshold= 4096
```

```
nfs_repeat_messages= 0
nfs_udp_duplicate_cache_size= 0
nfs_tcp_duplicate_cache_size= 5000
nfs_server_base_priority= 0
nfs_dynamic_retrans= 1
nfs_iopace_pages= 0
nfs_max_connections= 0
nfs_max_threads= 8
nfs_use_reserved_ports= 0
nfs_device_specific_bufs= 1
nfs_server_clread= 1
nfs_rfc1323= 0
nfs_max_write_size= 0
nfs_max_read_size= 0
nfs_allow_all_signals= 0
```

These configurable variables can be also be set with the **nfso** command. For example, to set the udpchecksum variable to its default value of 1, enter:

```
# nfso -d udpchecksum
```

If you change the nfsbuffer sizes, you must verify that the kernel variable sb_max is greater than the NFS buffer values chosen. The default value of sb_max is 1048576 on AIX 5L Version 5.1. If you need to increase the sb_max value, use the **no** command. Remember that everything changed with **no** or **nfso** is valid only until the next boot, unless these changes have been added to a boot script, for example, /etc/rc.nfs.

5.7.2 NFS client-side performance

The NFS client performance topic often concentrates on the number of biod daemons used. For biod daemons, there is a default number of biod daemons (six for a NFS Version 2 mount, four for a NFS Version 3 mount) that may operate on any one remote-mounted file system concurrently. The idea behind this limitation is that allowing more than a set number of biod daemons to operate against the server at one time may overload the server. Since this is Config_Rules on a per-mount basis on the client, adjustments can be made to configure client mounts by the server capabilities.

When evaluating how many biod daemons to run, you should consider the server capabilities as well as the typical NFS usage on the client machine. If there are multiple users or multiple processes on the client that will need to perform NFS operations to the same NFS-mounted file systems, you have to be aware that contention for biod services can occur with just two simultaneous read or write operations.

Up to six biod daemons can be working on reading a file in one NFS file system. If another read starts in another NFS-mounted file system, both reads will be attempting to use all six biod daemons. In this case, presuming that the server(s) are not already overloaded, performance will likely improve by increasing the biod number to 12. This can be done using the **chnfs** command:

```
# chnfs -b 12
```

On the other hand, suppose both file systems are mounted from the same server and the server is already operating at peak capacity. Adding another six biod daemons could dramatically worsen the response due to the server dropping packets and resulting in time-outs and retransmits.

5.7.3 Mount options

The **mount** command has several NFS-specific options that may affect performance.

The most useful options are used to set the read and write sizes to some value that changes the read/write packet size that is sent to the server.

For NFS Version 3 mounts, the read/write sizes can be both increased and decreased. The default read/write sizes are 32 KB. The maximum possible on AIX at the time of publication is 61440 bytes (60 x 1024). Using 60 KB, read/write sizes may provide slight performance improvement in specialized environments. To increase the read/write sizes when both server and client are AIX machines requires modifying settings on both machines. On the client, the mount must be performed by setting up the read/write sizes with the -o option. For example, -o rsize=61440,wsize=61440. On the server, the advertised maximum read/write size is configured through use of the **nfso** command using the nfs_max_write_size and nfs_max_read_size parameters. For example:

```
# nfso -o nfs_max_write_size=61440
```

NFS Version 3 uses TCP by default, while NFS Version 2 uses UDP only. This means the initial client mount request using TCP will fail. To provide backwards compatibility, the mount is retried using UDP, but this only occurs after a time-out of some minutes. To avoid this problem, NFS Version 3 provided the proto and vers parameters with the **mount** command. These parameters are used with the -o option to hardwire the protocol and version for a specific mount. The following example forces the use of UDP and NFS Version 2 for the mount request:

```
# mount -o proto=udp,vers=2,soft,retry=1 server4:/tmp /mnt
```

5.8 Command summary

The following section provides a list of the key commands discussed in this chapter.

5.8.1 The netstat command

The syntax of the **netstat** command is as follows.

To display active sockets for each protocol or routing table information:

```
/bin/netstat [ -n ] [ {  -A -a } |  { -r  -i  -I  interface } ]
[ -f  AddressFamily ] [ -p  protocol ] [ interval ] [ system ]
```

To display the contents of a network data structure:

```
/bin/netstat [ -m | -s |  -ss |  -u |  -v ] [  -f AddressFamily ]
[ -p  protocol ] [ interval ] [  system ]
```

To display the packet counts throughout the communications subsystem:

```
/bin/netstat -D
```

To display the network buffer cache statistics:

```
/bin/netstat -c
```

To display the data link provider interface statistics:

```
/bin/netstat -P
```

To clear the associated statistics:

```
/bin/netstat [ -Zc | -Zi | -Zm | -Zs ]
```

Some useful **netstat** flags from a NFS perspective are provided in Table 5-2.

Table 5-2 Commonly used flags of the netstat command

Flags	Description
-P *protocol*	Shows statistics about the value specified for the protocol variable
-s	Shows statistics for each protocol
-D	Shows the number of packets received, transmitted, and dropped in the communications subsystem

5.8.2 The tcpdump command

The syntax of the **tcpdump** command is:

```
tcpdump [ -I ] [ -n ] [ -N ] [ -t ] [ -v ] [ -c count ] [ -i interface ]
[ -w file ] [ Expression ]
```

Some useful **tcpdump** flags are provided in Table 5-3.

Table 5-3 Commonly used flags of the tcpdump command

Flags	Description
-c *count*	Exits after receiving count packets.
-n	Omits conversion of addresses to names.
-N	Omits printing domain name qualification of host names.
-t	Omits the printing of a timestamp on each dump line.
-i *interface*	Listens on *interface.*

5.8.3 The iptrace command

The syntax of the **iptrace** command is:

```
iptrace [ -a ] [ -e ] [ -Pprotocol ] [ -iinterface ] [ -pport ]
[ -shost [ -b ] ] [ -dhost [ -b ] ] LogFile
```

Some useful **iptrace** flags are provided in Table 5-4.

Table 5-4 Commonly used flags of the iptrace command

Flags	Description
-a	Suppresses ARP packets.
-s *host*	Records packets coming from the source host specified by the host variable.
-b	Changes the -d or -s flags to bidirectional mode.

5.8.4 The ipreport command

The syntax of the **ipreport** command is:

```
ipreport [ -e ] [ -r ] [ -n ] [ -s ] LogFile
```

Some useful **ipreport** flags are provided in Table 5-5 on page 201.

Table 5-5 Commonly used flags of the ipreport command

Flags	Description
-s	Prepends the protocol specification to every line in a packet
-r	Decodes remote procedure call (RPC) packets
-n	Includes a packet number to facilitate easy comparison of different output formats

5.9 Quiz

The following assessment questions help verify your understanding of the topics discussed in this chapter.

1. Which of the following commands is most useful in collecting data to determine a network performance problem?

   A. **`iostat`**

   B. **`lpstat`**

   C. **`netstat`**

   D. **`vmstat`**

2. Which of the following commands should be used to test name resolution response?

   A. **`host`**

   B. **`iostat`**

   C. **`ifconfig`**

   D. **`hostname`**

3. Which of the following commands should be used to identify the cause of packet sequence problems?

   A. **`tprof`** and **`gprof`**

   B. **`netstat`** and **`iostat`**

   C. **`lsattr`** and **`ifconfig`**

   D. **`iptrace`** and **`tcpdump`**

4. Which of the following commands should be used to make a difference when tuning NFS on a client system?

   A. mount

   B. vmtune

   C. exportfs

D. schedtune

5. On an NFS server, **netstat -m** is run once and then again 15 minutes later.

```
# date
Mon Jun 12 20:05:54 CDT 1995

# netstat -m
1385 mbufs in use:
96 mbuf cluster pages in use
2170 Kbytes allocated to mbufs
841 requests for mbufs denied
0 calls to protocol drain routines

Kernel malloc statistics:
By sizeinusecallsfailedfreehiwat freed
32 30127870 2116400
64 41 3180  23 3200
128117 18140  43 1600
256 1386 3911620223840
512 20 11120  20 40 0
1024 2  77 0  2  20 0
2048 0  33 0  2  10 0
4096 96 335617841111200
16384 1  0  12 24 7

# date
Mon Jun 12 20:20:04 CDT 1995

# netstat -m
1389 mbufs in use:
97 mbuf cluster pages in use
2180 Kbytes allocated to mbufs
982 requests for mbufs denied
0 calls to protocol drain routines

Kernel malloc statistics:
By sizeinusecallsfailedfreehiwatfreed
32 30127870 2116400
64 41 3180  23 3200
128117 18140  43 1600
256 1386 3911620223840
512 20 11120  20 40 0
1024 2  77 0  2  20 0
2048 0  33 0  2  10 0
4096 97 338610982111200
16384 1  0  12 24 7
```

An extract of the output for both times is shown in the preceding exhibits.

Using the **netstat** outputs, which of the following conclusions is most appropriate to draw?

A. The number of biod(s) should be increased.

B. The number of nfsd(s) should be increased.

C. The machine needs more physical memory.

D. The machine needs more memory allocated for mbufs.

6. There is a TCP/IP application that receives files from a remote machine. The application reads 32 kilobytes of data at a time to the socket, but has not issued a system call to set the window size. Which of the following procedures should be performed on the local machine to increase the throughput of the application?

   A. Increase the size of thewall.

   B. Increase the size of sb_max.

   C. Increase the size of tcp_recvspace.

   D. Increase the size of tcp_sendspace.

7. Which of the following tools should be used to determine network latency or delay?

   A. **arp**

   B. **netstat**

   C. **ifconfig**

   D. **traceroute**

8. When analyzing network I/O performance, which of the following commands should be run?

   A. **vmstat**

   B. **iostat**

   C. **netstat -s**

   D. **wlmstat**

9. Which **nfso** option will print a list of all configurable options and their current values?

   A. **nfso -l**

   B. **nfso -o**

   C. **nfso -d**

   D. **nfso -a**

5.9.1 Answers

The following are the preferred answers to the questions provided in this section.

1. C
2. A
3. D
4. A
5. D
6. C
7. D
8. C
9. D

5.10 Exercises

The following exercises provide sample topics for self study. They will help ensure comprehension of this chapter.

1. Capture data from a telnet session using the **tcpdump** command.
2. Capture data from a telnet session using the **iptrace** command.
3. Compare outputs from previously captured sessions.
4. Using the **no** command, check current values of kernel variables.
5. Check protocol statistics with the **netstat** command using the -p flag.

6

Performance management tools

In this chapter, the following topics are covered:

- The AIX scheduler
- The multiple run queue design of AIX Version 4.3.3
- The **schedtune** command
- The **nice** and **renice** commands
- Workload Manager introduction

The scope of this chapter concentrates on the thread scheduling and the possibilities to manipulate the process priorities with the **schedtune**, **nice**, and **renice** commands. The **ps**, **bindprocessor**, **emstat**, and **tprof** commands are also reviewed.

6.1 The AIX scheduler

The need for a scheduler for operating system efficiency is paramount. There are more threads and processes running than there are CPUs on any system. This is why the operating system is using the scheduler to decide which thread is allowed to use the CPU at any moment. The scheduler selects the thread to run from a list of waiting ready-to-run threads on the *run queue*. The number of waiting threads on the run queue is provided in the highlighted leftmost column in the following **vmstat** output:

```
# vmstat 2 5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 16272 75548   0   0   0   0    0   0 102   21  10  1  0 99  0
 2  1 16272 75547   0   0   0   0    0   0 407 1541  24 49  0 51  0
 2  1 16272 75547   0   0   0   0    0   0 405   58  28 50  0 50  0
 2  1 16272 75547   0   0   0   0    0   0 406   43  25 50  0 50  0
 2  1 16272 75547   0   0   0   0    0   0 409   29  26 50  0 50  0
```

The threads in the run queue are sorted in priority order, and the thread that has the highest priority gets to use the CPU. For more information on the relationship between process and threads, see Chapter 2, “Performance tuning: Getting started” on page 7.

In AIX Version 4, the five possible values for the thread scheduling policy are as follows:

SCHED_FIFO This is a non-preemptive scheduling scheme. After a thread with this policy is scheduled, it runs to completion unless it is blocked, it voluntarily yields control of the CPU, or a higher-priority thread becomes dispatchable. Only fixed-priority threads can have a SCHED_FIFO scheduling policy.

SCHED_RR The thread has a fixed priority. When a SCHED_RR thread has control at the end of the time slice, it moves to the tail of the queue of dispatchable threads of its priority. Only fixed-priority threads can have a SCHED_RR scheduling policy.

SCHED_OTHER This policy is defined by POSIX Standard 1003.4a as implementation-defined. In AIX Version 4, this policy is defined to be equivalent to SCHED_RR, except that it applies to threads with nonfixed priority. The recalculation of the running thread's priority at each clock interrupt means that a thread may lose control because its priority has risen above that of another dispatchable thread.

SCHED_FIFO2 This policy is the same as for SCHED_OTHER, except that it allows a thread that has slept for only a short amount of time to be put at the head of its run queue when it is awakened. This policy is only available on AIX Version 4.3.3 or later.

SCHED_FIFO3 A thread whose scheduling policy is set to SCHED_FIFO3 is always put at the head of a run queue. To prevent a thread belonging to the SCHED_FIFO2 scheduling policy from being put ahead of SCHED_FIFO3, the run queue parameters are changed when a SCHED_FIFO3 thread is enqueued, so that no thread belonging to SCHED_FIFO2 will join the head of the run queue. This policy is only available on AIX Version 4.3.3 or later.

6.1.1 Priority calculation on AIX versions prior to 4.3.2

The priority values differ from AIX versions previous to 4.3.2 and AIX Version 4.3.2 and later. Generally speaking, the lower the value is, the higher is the priority, with 0 as the lowest value and the greatest priority. The other end is the value of 127, which is the worst priority a thread can get. This priority value is reserved for the wait process. While the thread is running (using the CPU), the priority is recalculated, the value goes up, and the priority goes down. The longer a thread has existed without using CPU, the lower the value gets and, accordingly, the higher the priority. At one point, a thread in the run queue will have a lower value (higher priority) than the current running thread, the thread running is released, and the thread from the run queue is given CPU time.

As shown in Figure 6-1 on page 208, the global run queue used by AIX versions prior to 4.3.2 is symbolized. Threads A and B are forced to release control of the CPUs as soon as higher priority threads become runnable. In this case, threads C, D, and E are all available. Thread C is chosen first because it has the highest priority. Threads D and E have equal priority and occupy adjacent positions in the queue. Thread D is selected because of its position.

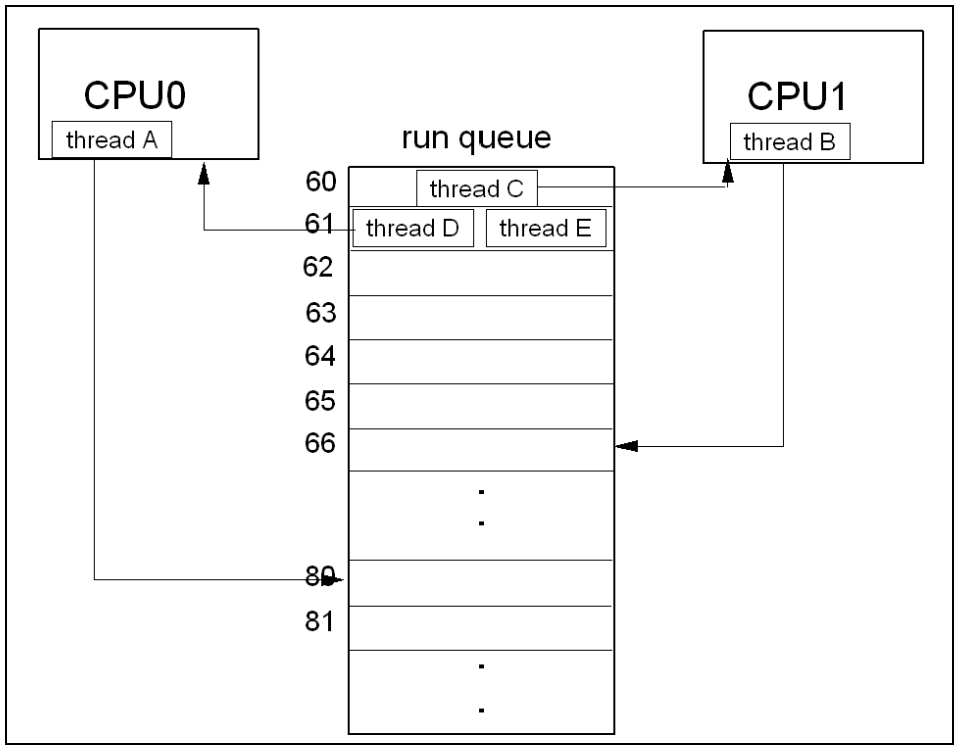


Figure 6-1 Run queue prior to AIX Version 4.3.3

Figure 6-1 is a simplification of the actual layout as shown in Figure 6-2 on page 209. All the dispatchable threads of a given priority occupy consecutive positions in the run queue. AIX Version 4 maintains 128 run queues. These run queues relate directly to the range of possible values (0 through 127) for the priority field for each thread. This method makes it easier for the scheduler to determine which thread is most favored to run. Without having to search a single large run queue, the scheduler consults a 128-bit mask where a bit is on to indicate the presence of a ready-to-run thread in the corresponding run queue.

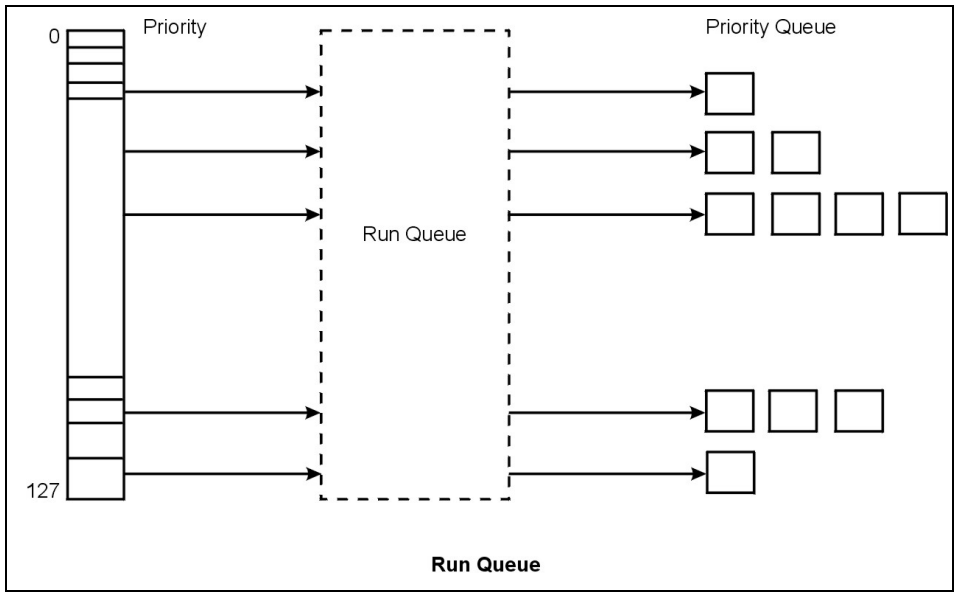


Figure 6-2 AIX Version 4, 128 run queues

A thread can be fixed priority or non-fixed priority. The priority value of a fixed-priority thread is constant, while the priority value of a non-fixed priority thread is the sum of the maximum priority level for user threads (a constant 40), the thread's nice value (the default is 20 for foreground processes and 24 for background processes), and its CPU penalty.

One of the factors in priority calculation is the *recent CPU usage* value. One out of two calculations used in defining the recent CPU usage is:

```
Recent CPU usage = Old recent CPU usage + 1
```

This calculation is done 100 times a second (at every tick). The recent CPU usage value increases by 1 each time the thread is in control of the CPU at the end of a tick. The maximum value is 120. In other words, running threads have their recent CPU usage value recalculated (increased) 100 times a second until reaching the maximum limit of 120. This value is shown in the C column of the **ps** command output:

```
# ps -f
     UID   PID  PPID   C    STIME    TTY  TIME CMD
    root 12948 12796   0 14:27:07  pts/1  0:00 ksh
    root 13888 12948 111 10:08:34  pts/1 94:21 ./tctestprg
    root 15432 12948   4 11:42:56  pts/1  0:00 ps -f
    root 15752 12948 110 10:08:34  pts/1 94:21 ./tctestprg
```

Once every second, all threads, including those that are asleep, have their recent CPU usage value recalculated as follows:

```
Recent CPU usage = Old recent CPU usage x (SCHED_D / 32)
```

The default value for SCHED_D is 16, which means that the old recent CPU usage value is divided by 2 (16 / 32 = 0.5). This prevents the recent CPU usage values of all processes from being set at a stable 120.

With this value, recent CPU usage, the CPU penalty value can be calculated:

```
CPU penalty = Recent CPU usage x (SCHED_R / 32)
```

The default value for SCHED_R is 16. With the CPU penalty value defined, the *priority*, also calculated at every tick, can finally be calculated as follows:

```
Priority value = 40 + nice value + CPU penalty
```

In this calculation, the default nice value is 20 for foreground processes and 24 for background processes.

With these definitions, you can see that three values can be manipulated for tuning performance: the nice value, SCHED_R (also called the weighting factor), and the SCHED_D (also called the decay factor).

6.1.2 Priority calculation on AIX Version 4.3.2 and later

AIX Version 4.3.2 and later added a couple of new definitions to be considered. First is the NICE factor, which is not the nice value manipulated with the **nice** command, but the sum of the maximum priority level for user threads and the value manipulated by the **nice** command.

Secondly, the DEFAULT_NICE factor is added to the algorithm. This factor is equal to the maximum priority level for a user, also named the base value (40), plus the default nice value for a foreground process (20). In other words, the sum is 60 for a foreground process (DEFAULT_NICE - NICE = 60).

The following calculation is used for calculating the priority:

```
Priority = (Recent CPU usage x SCHED_R x (xnice + 4)) / (32 x (DEFAULT_NICE +
4)) + xnice
```

Where DEFAULT_NICE = 40 + 20 (base value plus default nice). The calculation for the xnice value is as follows:

```
xnice = (NICE > DEFAULT_NICE) ? (2 * nice) - 60 : NICE
```

This means that if nice is smaller or equal to DEFAULT_NICE, then:

```
xnice = NICE
```

But if NICE is greater than DEFAULT_NICE (in other words, if you have manipulated the thread with the **nice** command to lessen its priority), then:

```
xnice = (2 x NICE) - 60
```

The nice value has a much greater impact on the priority of a thread. It is now included in the calculation as a multiple of the recent CPU usage, in addition to its use as a constant factor. To get greater granularity in the run queue(s), the DEFAULT_NICE is set to 60. Some artificial values will help show the calculation.

```
Recent CPU usage = 64
SCHED_R = 16
NICE = 64
```

Starting with the XNICE calculation:

```
xnice = (NICE > DEFAULT_NICE) ? (2 * NICE) - 60 : nice
```

Because NICE is greater then DEFAULT_NICE, then:

```
xnice = (2 x 64) - 60 = 68
```

By entering the values given and the XNICE value in the calculation:

```
Priority = (Recent CPU usage x SCHED_R x (xnice + 4)) / (32 x (DEFAULT_NICE +
4)) + xnice
```

The calculation will look as follows:

```
P = (64 x 16 x (68 + 4)) / (32 x 64) + xnice
P = (73728 / 2048) + 64
P = 100
```

You still have three values to manipulate: The nice value (as in the example), SCHED_R, and SCHED_D (for recent CPU usage). In the following sections, the multiple run queue layout and the commands that are used to change these values are discussed.

6.2 Multiple run queues with load balancing

The run queue is the same global queue on AIX Version 4.3.2 as on AIX Version 4.3.1, but on AIX Version 4.3.3 and later, the run queue layout has changed. AIX now offers improved cache affinity through the use of multiple run queues. The new kernel scheduler implements a single global run queue along with a set of local run queues, where each processor has a dedicated local run queue.

Once a thread is placed on a local run queue, it generally stays there until an imbalance is detected. Thresholds are used to limit the amount of load balancing that takes place.

A diagram of the relationship of the local and global run queues is provided in Figure 6-3.

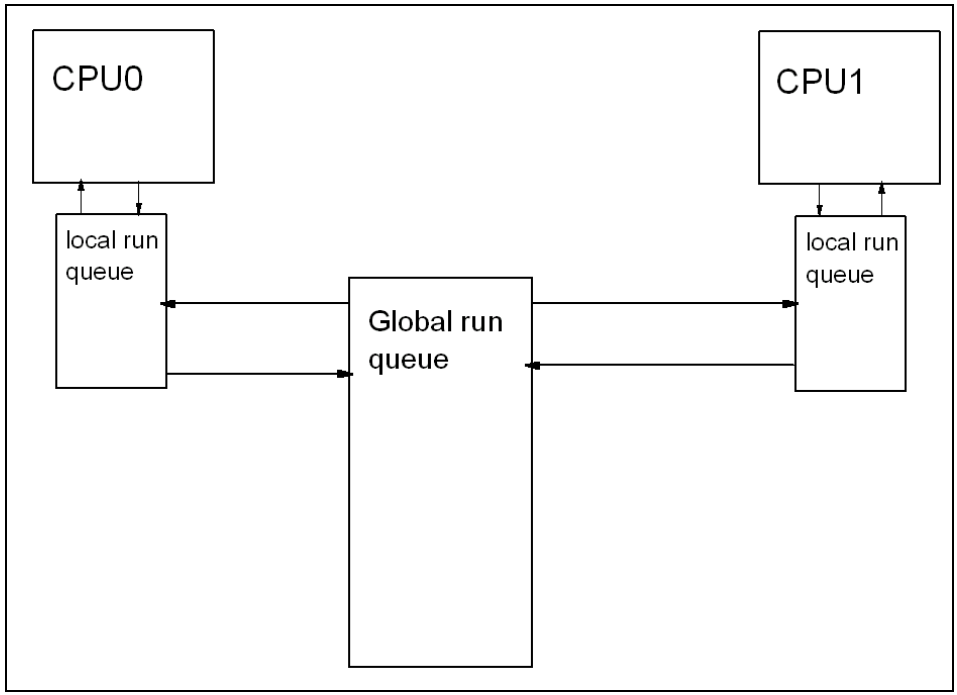


Figure 6-3 Run queue on AIX Version 4.3.3 and later

The per-node local run queue competes with the local run queues of the node for CPUs to service its threads. The priority of the highest thread on a run queue (both local and global run queues) is maintained in the run queue. The dispatcher uses this data without holding the run queue locks to make a low overhead decision of which run queue to search. This mechanism allows the priorities of the two run queues to be honored most of the time. When both run queues have threads waiting at the same priority, the local run queue is chosen.

Usually, when initiated, threads get on the global run queue by means of the load balancing mechanism implemented. Once a CPU dispatches a thread from the global run queue, it does not generally return to the global run queue, but rather to the queue served by the CPU dispatching it.

The load balancing is handled by a number of algorithms designed to keep the various run queues of a system approximately equally utilized. There are four balancing algorithms, which are discussed in the following sections.

6.2.1 Initial load balancing

Initial load balancing applies to newly created threads. When an unbound new thread is created as part of a new process (as well as a new thread for an existing process), it is placed on an idle CPU (if one exists). If an idle CPU cannot be found, the thread will be placed in the global queue.

6.2.2 Idle load balancing

Idle load balancing applies when a process would otherwise go idle, running the *waitproc* thread (for example, PID 516). When the dispatcher reaches this point in its logic, it does not just scan other run queues in an attempt to find work at any cost. It is actually beneficial to allow what appears to be unnecessary idle cycles rather than moving a thread and losing cache affinity. The steps taken by the idle load balancing method are:

- Before dispatching the waitproc, search other queues for *available* work. This is a stronger statement than work *being* on another queue. The search routine will look for a queue that:
  - Contains the largest number of runnable threads.
  - Contains more runnable threads than the current *steal threshold.*
  - Contains at least one stealable (unbound) thread.
  - Has not had *steal_max* threads already stolen from it over the current clock tick interval.

  The search is done without holding those run queues' locks.
- To actually steal a thread, the chosen run queue's lock must be obtained. This is done by a special call written to avoid interfering with another instance of the dispatcher. If no lock can be obtained, run the waitproc.
- After getting the lock, check that a stealable thread is still available. If there is no stealable thread, the waitproc is run.
- Change the threads' run queue assignment and pointer.

6.2.3 Frequent periodic load balancing

This is performed every N clock ticks (at time of publication, 10). It attempts to balance the loads on the local queues of a node in much the same way that idle load balancing does. The idea is to move a thread from the most loaded to the

least loaded run queue, but if the least loaded run queue has stolen a thread through idle load balancing in the last interval, nothing is done. The difference in load factors between the two run queues chosen for frequent periodic load balancing must be at least 1.5. Idle load balancing is less expensive, so the ideal situation is if frequent periodic load balancing does not have to interfere.

6.2.4 Infrequent periodic load balancing

If a thread has not received CPU time in the last N.5 (at time of publication 1.5) seconds, the thread is moved to the global run queue.

6.3 Scheduler performance management

AIX offers several ways to modify the default behavior of the scheduling system for threads and processes. This section describes the **schedtune**, **nice**, and **renice** commands.

6.3.1 The schedtune command

The **schedtune** command allows you to specify the SCHED_R with the -r flag and the SCHED_D with the -d flag. When executing the **schedtune** command without flags, the current values will be shown:

```
# /usr/samples/kernel/schedtune

      THRASH           SUSP       FORK             SCHED
-h     -p    -m      -w    -e      -f       -d       -r        -t       -s
SYS   PROC  MULTI   WAIT  GRACE   TICKS SCHED_D  SCHED_R TIMESLICE MAXSPIN
 0    4     2        1      2      10       16       16         1        16384

   CLOCK
   -c
%usDELTA
   100
```

Tuning is accomplished through two flags of the **schedtune** command: -r and -d. Each flag specifies a parameter that is an integer from 0 to 32. The parameters are applied by multiplying the recent CPU usage value by the parameter value and then dividing by 32. The default SCHED_R and SCHED_D values are 16, as seen in the previous output. The following sections discuss different uses of the **schedtune** command.

Schedtune example one

The following command will result in SCHED_R = 0 and SCHED_D = 0.5:

```
# /usr/samples/kernel/schedtune -r 0
```

This would mean that the CPU penalty was always 0, making priority absolute. No background process would get any CPU time unless there were no dispatchable foreground processes at all, as background processes in **ksh** are started with adding 4 to the nice value of the parent shell. The priority values of the threads would effectively be constant, although they would not technically be fixed-priority threads.

Schedtune example two

The following command would result in an effective SCHED_R = 1 and SCHED_D = 1:

```
# /usr/samples/kernel/schedtune -r 32 -d 32
```

This would mean that long-running threads would reach a C value of 120 and remain there, contending on the basis of their nice values. New threads would have priority, regardless of their nice value, until they had accumulated enough time slices to bring them within the priority value range of the existing threads.

Schedtune example three

The most likely reason to manipulate the values would be to make sure that background processes do not compete with foreground processes. By making SCHED_R smaller, you can restrict the range of possible priority values. For example:

```
# /usr/samples/kernel/schedtune -r 5
```

(SCHED_R = 0.15625, SCHED_D = 0.5) would mean that a foreground process would never have to compete with a background process started with the command **nice -n 20**. The limit of 120 CPU time slices accumulated would mean that the maximum CPU penalty for the foreground process would be 18. In Figure 6-4 on page 216, this relationship is graphically shown. Because the CPU penalty will get a maximum value of 18, the foreground process with a nice value of 20 will always, when it needs, get CPU. On the other hand, the background process, with a nice value of 40, will use CPU only when the foreground process does not need the CPU.

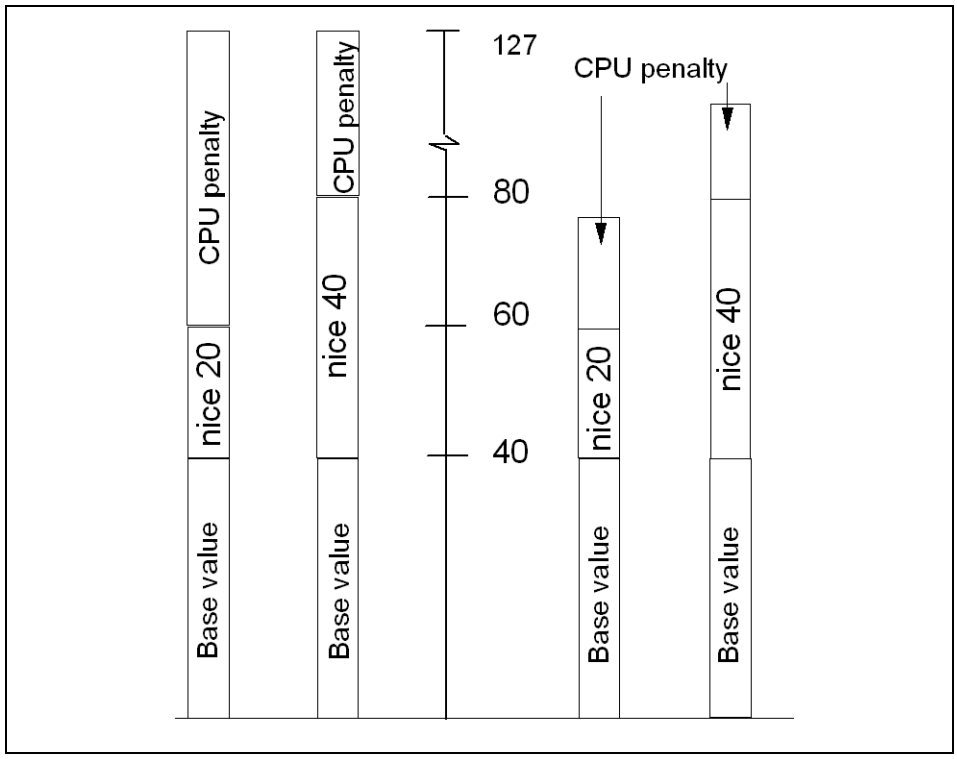


Figure 6-4 CPU penalty example

SCHED_R and SCHED_D guidelines

The following discusses guidelines for tuning performance using SCHED_R and SCHED_D.

- Smaller values of SCHED_R narrow the priority range and the nice value has more of an impact on the priority.
- Larger values of SCHED_R widen the priority range and the nice value has less of an impact on the priority.
- Smaller values of SCHED_D decay CPU usage at a faster rate and can cause CPU-intensive threads to be scheduled sooner.
- Larger values of SCHED_D decay CPU usage at a slower rate and penalize CPU-intensive threads more (thus favoring interactive-type threads).

If you conclude that one or both parameters need to be modified to accommodate your workload, you can enter the **schedtune** command while logged on as root user. The changed values will persist until the next **schedtune** command that modifies them, or until the next system boot. Values can be reset

to their defaults with the command **schedtune -D**, but remember that all **schedtune** parameters are reset by that command, including VMM memory load control parameters. To make a change to the parameters that will persist across boots, add an appropriate line at the end of the /etc/inittab file (or in the AIX 5L Version 5.2 /etc/tunables file.

VMM memory load control using schedtune

The VMM memory load control facility protects an overloaded system from thrashing. For early versions of the operating system, if a large number of processes hit the system at the same time, memory became overcommitted and thrashing occurred, causing performance to degrade rapidly. A memory-load control mechanism was developed that could detect thrashing. Certain parameters affect the function of the load-control mechanism.

With the **schedtune** command, the root user can affect the criteria used to determine thrashing, the criteria used to determine which processes to suspend, the length of time to wait after thrashing ends before reactivating processes, the minimum number of processes exempt from suspension, or reset values to the defaults.

Scheduler tuning using schedtune

With the **schedtune** command you can change scheduler parameters at runtime, and these changes will be cleared at the next reboot. If you find usefull settings for the system, then run **schedtune** in rc.local. The average time slice of the scheduler can be changed with the -tN parameter. The system uses 1 for N as the default. Setting N to any other value will change the maximum number of clock ticks between dispatches of processes. Usually the scheduler dispatches processes once per clock interrupt (roughly every 10 ms), so the maximum time a process can be active before the dispatcher tries to dispatch another one is 10 ms.

Other **schedtune** parameters enable you to set the delay before retrying a failed fork function, to set the delay after thrashing before a process gets into the run queue again, and to set how long a process has to be able to run before being suspended.

Note: The **schedtune** command is in the samples directory because it is very VMM-implementation dependent. The **schedtune** code that accompanies each release of the operating system is tailored specifically to the VMM in that release. Running the **schedtune** command from one release on a different release might result in an operating-system failure. It is also possible that the functions of the **schedtune** command may change from release to release. Do not propagate shell scripts or /etc/inittab entries that include the **schedtune** command to a new release without checking the **schedtune** documentation for the new release to make sure that the scripts will still have the desired effect.

The **schedtune -?** command provides a description of the flags and options. The source and object code of the **schedtune** command are in /usr/samples/kernel.

The schedtune command summary

The **schedtune** command is used to manipulate the scheduler and the swapper. There are some major differences between the commands in AIX Version 4.3.2 and AIX Version 4.3.3.

The syntax of the **schedtune** command is:

```
schedtune [ -D | { [ -d n ] [ -e n ] [ -f n ] [ -h n ] [ -m n ] [ -p n ]
[ -r n ] [ -t n ] [ -w n] } ]
```

Table 6-1 gives some useful **schedtune** flags from a CPU-tuning perspective.

Table 6-1 Commonly used flags of the schedtune command

Flag	Description
-r	Manipulates the SCHED_R weighting factor
-d	Manipulates the SCHED_D decay factor
-D	Resets all **schedtune** values to default values

6.3.2 The nice and renice commands

The nice value has been explained in “The nice and renice commands” on page 15. The nice value can be seen with the **ps** command in the NI column:

```
$ ps -lu thomasc
      F S   UID   PID  PPID   C PRI NI ADDR    SZ TTY  TIME CMD
 200001 A 15610  5204 15476   3  61 20 a655   344 pts/1  0:00 ps
 200001 A 15610 15476 12948   1  60 20 5029   488 pts/1  0:00 ksh
 200001 A 15610 15818 15476 120 126 24 408b    44 pts/1  0:25 tctest
 200001 A 15610 16792 15476 120 126 24 e89e    44 pts/1  0:18 tctest
```

Two programs have been started in the background, as shown by the nice value of 24, while the **ps** command running in the foreground has a nice value of 20. All outputs from the **ps** command have been edited in this to fit the screen.

Running a command with nice

Any user can run a command to set a less-favorable-than-normal priority by using the **nice** command. Only the root user can use the **nice** command to run commands at a more-favorable-than-normal priority. From the root user, the **nice** command values range between -20 and 19.

With the **nice** command, the user specifies a value to be added to or subtracted from the default nice value. The modified nice value is used for the process that runs the specified command. The priority of the process is still non-fixed; that is, the priority value is still recalculated periodically based on the CPU usage, nice value, and minimum user-process-priority value.

The user thomasc, not being the root user, has the a range of nice values between 1 and 19 available. When applying a nice value to a background command, note that the maximum NI value is 39 even though the calculated value would be 43 (34 + 9), as shown in the following example:

```
# id
uid=15610(thomasc) gid=0(system)
# nice -19 ./tprof.tctestprg &
# ps -al|head -1 ; ps -al |grep tctestprg
       F S   UID   PID  PPID   C PRI NI ADDR    SZ TTY  TIME CMD
  200001 A 15610 14740 15490  90 126 39 5888    44 pts/3  0:58 tctestprg
  240001 A 15610 15818     1  90 118 24 408b    44 pts/1 51:02 tctestprg
  240001 A 15610 16792     1  89 118 24 e89e    44 pts/1 50:55 tctestprg
```

The root user has the possibility to lessen the nice value. Notice the syntax; the first dash (-) is only an option marker, and the other dash tells the **nice** command to subtract 15 from the default value of 24 (the process is started in the background). For example:

```
# nice --15 ./tprof/tctestprg &
# ps -al|head -1 ; ps -al |grep tctestprg
       F S   UID   PID  PPID   C PRI NI ADDR    SZ TTY   TIME      CMD
  200001 A 15610 14740 15490  91 126 39 5888    44 pts/3 4:37  tctestprg
  240001 A 15610 15818     1  92 119 24 408b    44 pts/1 54:41 tctestprg
  200001 A 0    16304 12948  85 84  9 c0bb      44 pts/1 0:03 tctestprg
  240001 A 15610 16792     1  92 59 -- e89e    44 pts/1  54:34 tctestprg
```

Another way to use the **nice** command to get the same result as in the previous example, is with the -n flag, as follows:

```
# nice -n -15 ./tprof/tctestprg &
```

It is actually only in this case, where root lessens the nice value, that a significant change is seen in the priority value. In the preceding output, the values nice=39 and nice=24 generate similar priority values (PRI column), but process 16304, started by root with nice=9, has a significant advantage with a priority value of 84.

The output also shows the scenario when a process is executed with a fixed priority (PID 16792). In the PRI column, the set priority is shown to be - 59, and the NI column shows no value. This can be done with the setpri subroutine. The setpri subroutine sets the scheduling priority of all threads in a process to be a constant.

Changing the nice value on a running thread

The **renice** command, which has a similar syntax as the **nice** command, allows you to modify the nice value on a running process. The example from the previous section is used. For example, subtract 5 from the actual value of 9. On the tctestprg, root is running:

```
# renice -n -5 16304
# ps -al|head -1 ; ps -al |grep tctestprg
F      S   UID   PID  PPID   C PRI NI ADDR    SZ TTY  TIME CMD
200001 A 15610 14740 15490  94 126 39 5888    44 pts/3 17:13 tctestprg
240001 A 15610 15818     1  94 120 24 408b    44 pts/1 67:17 tctestprg
200001 A     0 16304 12948  86  76  4 c0bb    44 pts/1 12:37 tctestprg
240001 A 15610 16792     1  94 120 24 e89e    44 pts/1 67:10 tctestprg
```

The PID is used to identify which program (or more correctly which thread) is to be manipulated.

The nice and renice commands summary

The **nice** and **renice** commands are used to manipulate the nice value for the threads of a process.

The syntax of the **nice** command is:

```
nice [ - Increment| -n Increment ] Command [ Argument ... ]
```

Some commonly used **nice** flags are listed in Table 6-2.

Table 6-2 Commonly used flags of the nice command

Flags	Description
-*increment*	Increments a command's priority up or down, by specifying a positive or negative number
-n*increment*	Same as previous flag

The syntax of the **renice** command is:

```
renice [ -n Increment ] [ -g | -p | -u ] ID ...
```

Some commonly used **renice** flags are listed in Table 6-2 on page 220.

Table 6-3 Commonly used flags of the renice command

Flags	Description
-n *increment*	Specifies the number to add to the nice value of the process. The value of *increment* can only be a decimal integer from -20 to 20.
-u *username user numeric ID*	Changes nice values for user.

Important: Do not manipulate the scheduler without a thorough understanding of the mechanisms controlling the scheduler.

6.4 The bindprocessor command

The **bindprocessor** command binds or unbinds the kernel threads of a process or it lists available processors. To bind or unbind threads, it requires two parameters, as follows:

```
bindprocessor Process ProcessorNum
```

The process parameter is the process identifier of the process whose threads are to be bound or unbound, and the ProcessorNum parameter is the logical processor number of the processor to be used. If the ProcessorNum parameter is omitted, the process is bound to a randomly selected processor.

A process itself is not bound, but rather its kernel threads are bound. Once kernel threads are bound, they are always scheduled to run on the chosen processor, unless they are later unbound. When a new thread is created, it has the same bind properties as its creator. This applies to the initial thread in the new process created by the fork subroutine: The new thread inherits the bind properties of the thread that called the fork. When the exec subroutine is called, thread properties are left unchanged.

To check what processors are available, enter the following command:

```
# bindprocessor -q
The available processors are:  0 1 2 3
```

To bind process 16792 to processor 2, enter the following command:

```
# bindprocessor 16792 2
```

To check which process threads are bound to which processor, check the BND column in the **ps** command output:

```
# ps -mo THREAD
USER   PID  PPID       TID ST   CP  PRI SC  F        TT  BND COMMAND
    root 12948 12796      - A    0  60  1 240001  pts/1   - ksh
       -     -     -   7283 S    0  60  1    400      -   - -
    root 13704 12948      - A    3  61  1 200001  pts/1   - ps -mo THREAD
       -     -     -  19391 R    3  61  1      0      -   - -
 thomasc 15818     1      - A   79 112  0 240001  pts/1   -./tprof/tctestprg
       -     -     -  16077 R   79 112  0      0      -   - -
    root 16304 12948      - A   77  72  0 200001  pts/1   -./tprof/tctestprg
       -     -     -  17843 R   77  72  0      0      -   - -
 thomasc 16792     1      - A   79 112  0 240001  pts/1   2./tprof/tctestprg
       -     -     -  16357 R   79 112  0      0      -   2 -
```

6.5 The vmtune command

The **vmtune** command changes operational parameters of the Virtual Memory Manager (VMM) and other AIX components. The command and source for **vmtune** are found in the /usr/samples/kernel directory. It is installed with the bos.adt.samples fileset.

The **vmtune** command syntax is as follows:

```
vmtune [ -b numfsbuf ] [ -B numpbuf ] [ -c numclust ] [ -f minfree ]
[ -F maxfree ] [ -k npskill ] [ -l lrubucket ] [ -M maxpin ] [ -N pd_npages ]
[ -p minperm ] [ -P maxperm ] [ -r minpgahead ] [ -R maxpgahead ]
[-u lvm_budcnt] [ -w npswarn ] [-W maxrandwrt]
```

An example of the **vmtune** command without any flags is shown in the following:

```
# /usr/samples/kernel/vmtune
vmtune:  current values:
  -p       -P        -r          -R         -f       -F       -N        -W
minperm  maxperm  minpgahead maxpgahead  minfree  maxfree  pd_npages maxrandwrt
  26007   104028       2          8        120      128     524288        0

  -M      -w      -k      -c        -b         -B           -u        -l    -d
```

```
maxpin npswarn npskill numclust numfsbufs hd_pbuf_cnt lvm_bufcnt lrubucket
defps
104851    4096    1024       1      93         80          9      131072     1

        -s              -n         -S           -h
sync_release_ilock  nokillroot  v_pinshm  strict_maxperm
        0               0           0             0

number of valid memory pages = 131063   maxperm=79.4% of real memory
maximum pinable=80.0% of real memory    minperm=19.8% of real memory
number of file memory pages = 102029    numperm=77.8% of real memory
```

The Virtual Memory Manager (VMM) maintains a list of free real-memory page frames. These page frames are available to hold virtual-memory pages needed to satisfy a page fault. When the number of pages on the free list falls below that specified by the minfree parameter, the VMM begins to steal pages to add to the free list. The VMM continues to steal pages until the free list has at least the number of pages specified by the maxfree parameter.

If the number of file pages (permanent pages) in memory is less than the number specified by the minperm parameter, the VMM steals frames from either computational or file pages, regardless of re-page rates. If the number of file pages is greater than the number specified by the maxperm parameter, the VMM steals frames only from file pages. Between the two, the VMM normally steals only file pages, but if the repage rate for file pages is higher than the repage rate for computational pages, computational pages are stolen as well.

If a process appears to be reading sequentially from a file, the values specified by the minpgahead parameter determine the number of pages to be read ahead when the condition is first detected. The value specified by the maxpgahead parameter sets the maximum number of pages that will be read ahead, regardless of the number of preceding sequential reads.

The **vmtune** command can only be executed by root. Changes made by the **vmtune** command last until the next reboot of the system. If a permanent change in VMM parameters is needed, an appropriate **vmtune** command should be put in /etc/inittab.

Table 6-4 on page 224 lists the flags and some of the limitations for the settings.

Table 6-4 Commonly used flags of the vmtune command

Flag	Description
-b *numfsbuf*	Specifies the number of file system bufstructs. The current default in AIX Version 4 is 93 because it is dependent on the size of the bufstruct. This value must be greater than 0. Increasing this value will help write performance for very large write sizes (on devices that support very fast writes). In order to enable this value, you must unmount the file system, change the value, and then mount the system again.
-B *numpbuf*	Controls the number of pbufs available to the LVM device driver. pbufs are pinned memory buffers used to hold I/O requests related to a journaled file system (JFS). On systems where large amounts of sequential I/O occurs, this can result in I/O bottlenecks at the LVM layer waiting for pbufs to be freed. In AIX Version 4, a single pbuf is used for each sequential I/O request regardless of the number of pages in that I/O. The maximum value is 128.
-c *numclust*	Specifies the number of 16 KB clusters processed by write behind. The default value is 1. Values can be any integer above 0. Higher number of clusters may result in larger sequential write performance on devices that support very fast writes (RAID and so on). Setting the value to a very high number, such as 500000, will essentially defeat the write-behind algorithm. This can be beneficial in cases such as database index creations, where pages that were written to are read a short while later; write-behind could actually cause this process to take longer. One suggestion is to turn off write-behind before building indexes and then turn it back on after indexes have been built. For example, the **mkpasswd** command can run significantly faster when write-behind is disabled.
-f *minfree*	Specifies the minimum number of frames on the free list. This number can range from 8 to 204800. The default value depends on the amount of RAM on the system. minfree is by default the value of maxfree: 8. The value of maxfree is equal to minimum (the number of memory pages divided by 128). The delta between minfree and maxfree should always be equal to or greater than maxpgahead.
-F *maxfree*	Specifies the number of frames on the free list at which page stealing is to stop. This number can range from 16 to 204800, but must be greater than the number specified by the minfree parameter by at least the value of maxpgahead.

Flag	Description
-k *npskill*	Specifies the number of free paging-space pages at which AIX begins killing processes. The formula to determine the default value of npskill in AIX Version 4 is: `MAX(64, number_of_paging_space_pages/128)` The npskill value must be greater than 0 and less than the total number of paging space pages on the system. The default value is 128.
-l *lrubucket*	This parameter specifies the size (in 4 KB pages) of the least recently used (LRU) page replacement bucket size. This is the number of page frames that will be examined at one time for possible page outs when a free frame is needed. A lower number will result in lower latency when looking for a free frame but will also result in behavior that is not as similar to a true LRU algorithm. The default value is 512 MB and the minimum is 256 MB. Tuning this option is not recommended.
-M *maxpin*	Specifies the maximum percentage of real memory that can be pinned. The default value is 80 percent. If this value is changed, the new value should ensure that at least 4 MB of real memory will be left unpinned for use by the kernel. maxpin values must be greater than 1 and less than 100. The value under maxpin is converted to a percentage at the end of the output of **vmtune**.
-N *pd_npages*	Specifies the number of pages that should be deleted in one chunk from RAM when a file is deleted. Changing this value may only be beneficial to real-time applications that delete files. By reducing the value of pd_npages, a real-time application can get better response time since a fewer number of pages will be deleted before a process or thread is dispatched. The default value is the largest possible file size divided by the page size (currently 4096); if the largest possible file size is 2 GB, then pd_npages is, by default, 524288.
-p *minperm*	Specifies the point below which file pages are protected from the repage algorithm. This value is a percentage of the total real-memory page frames in the system. The specified value must be greater than or equal to 5. The default value of the minperm percentage is always around 17–19 percent of memory.

Flag	Description
-P *maxperm*	Specifies the point above which the page stealing algorithm steals only file pages. This value is expressed as a percentage of the total real-memory page frames in the system. The specified value must be greater than or equal to 5. The default value of the maxperm percentage is always around 75–80 percent of memory. A pure Network File System (NFS) server may obtain better performance by increasing the maxperm value. A system that accesses large files (over 50–75 percent of the amount of RAM on the system; look at numperm to see how much memory is currently used for file mapping) may benefit by increasing the maxperm value. maxperm can be reduced on systems with large active working storage requirements (the AVM column from **vmstat** compared to total real page frames) to reduce or eliminate page space I/O.
-r *minpgahead*	Specifies the number of pages with which sequential read-ahead starts. This value can range from 0 to 4096. It should be a power of 2. The default value is 2.
-R *maxpgahead*	Specifies the maximum number of pages to be read ahead. This value can range from 0 to 4096. It should be a power of 2 and should be greater than or equal to minpgahead. The default value is 8. Increasing this number will help large sequential read performance. Because of other limitations in the kernel and the Logical Volume Manager (LVM), the maximum value should not be greater than 128. The delta between minfree and maxfree should always be equal to or greater than maxpgahead.
-u *lvm_bufcnt*	Specifies the number of LVM buffers for raw physical I/Os. The default value is 9. The possible values can range between 1 and 64. It may be beneficial to increase this value if you are doing large raw I/Os (that is, not going through the JFS).
-w *npswarn*	Specifies the number of free paging-space pages at which AIX begins sending the SIGDANGER signal to processes. The formula to determine the default value is: `MAX(512, 4*npskill)` The value of npswarn must be greater than 0 and less than the total number of paging space pages on the system. The default value is 512.

Flag	Description
-W *maxrandwrt*	Specifies a threshold (in 4 KB pages) for random writes to accumulate in RAM before these pages are syncd to disk using a write-behind algorithm. This threshold is on a per file basis. The -W maxrandwrt option is only available in AIX Version 4.1.3 and later. The default value of maxrandwrt is 0, which disables random write-behind. By enabling random write-behind (a typical value might be 128), applications that make heavy use of random writes can get better performance due to less dependence on the sync daemon to force writes out to disk. Some applications may degrade their performance due to write-behind (such as database index creations). In these cases, it may be beneficial to disable write-behind before creating database indexes and then re-enabling write-behind after the indexes are created.

6.6 Workload Manager (WLM)

WLM is designed to give system administrators more control over how the scheduler and the virtual memory manager (VMM) allocate resources to processes. This can be used to prevent different classes of jobs from interfering with each other and to allocate resources based on the requirements of different groups of users.

With WLM, you can create different classes of service for jobs, as well as specify attributes for those classes. These attributes specify minimum and maximum amounts of CPU, physical memory, and disk I/O throughput to be allocated to a class. WLM then assigns jobs automatically to classes using class assignment rules provided by a system administrator. These assignment rules are based on the values of a set of attributes for a process. Either the system administrator or a privileged user can also manually assign jobs to classes, overriding the automatic assignment.

The system administrator can specify the properties for the WLM subsystem by using either the Web-based System Manager graphical user interface, SMIT ASCII-oriented interface, the WLM command line interface, or by creating flat ASCII files. The Web-based System Manager and SMIT interfaces use the WLM commands to record the information in the same flat ASCII files. These files are named as follows:

classes Class definitions

description Configuration description text

limits Class limits

shares Class target shares

rules Class assignment rules

These files are called the WLM property files. A set of WLM property files defines a WLM configuration. You can create multiple sets of property files, defining different configurations of workload management. These configurations are located in subdirectories of /etc/wlm. The WLM property files describing the superclasses of the Config configuration are the file's classes, description, limits, shares, and rules in /etc/wlm/Config. Then, the property files describing the subclasses of the superclass super of this configuration are the file's classes, limits, shares and rules in directory /etc/wlm/Config/Super. Only the root user can start or stop WLM, or switch from one configuration to another.

The command to submit the WLM property files, **wlmcntrl**, and the other WLM commands allow users to specify an alternate directory name for the WLM properties files. This allows you to change the WLM properties without altering the default WLM property files.

A symbolic link, /etc/wlm/current, points to the directory containing the current configuration files. Update this link with the **wlmcntrl** command when you start WLM with a specified set of configuration files. The sample configuration files shipped with the operating system are in /etc/wlm/standard.

WLM configuration is performed through the preferred interface, the Web-based System Manager (Figure 6-5 on page 229), through a text editor and AIX commands, or through the AIX administration tool SMIT.

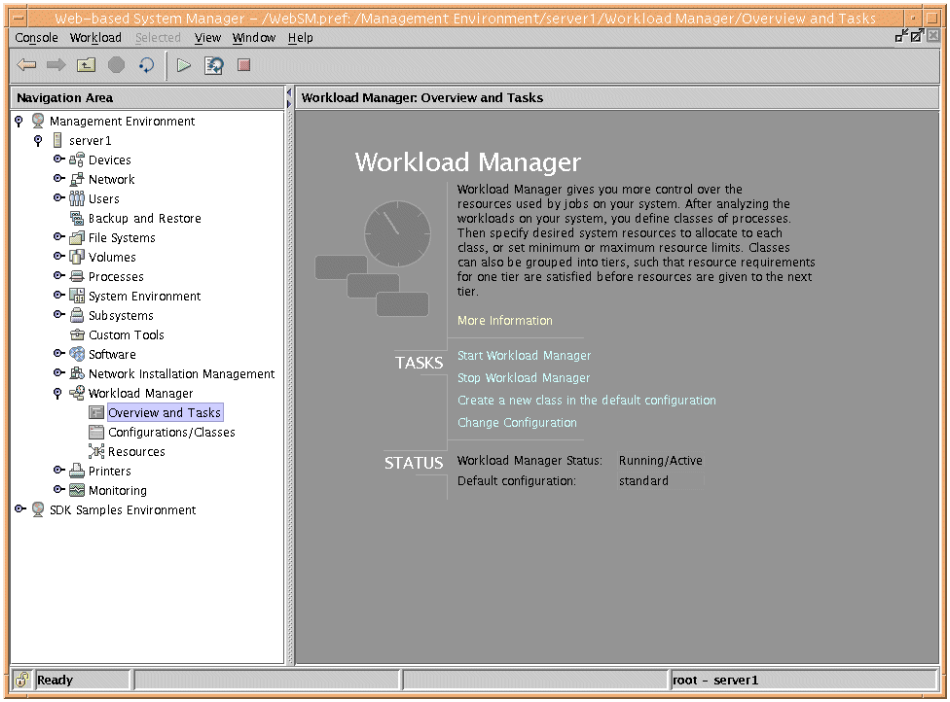


Figure 6-5 Web-based System Manager Overview and Tasks dialog

6.6.1 WLM concepts and architecture

The following section outlines the concepts provided with WLM on AIX 5L.

Classes

The central concept of WLM is the class. A class is a collection of processes (jobs) that has a single set of resource limits applied to it. WLM assigns processes to the various classes and controls the allocation of system resources among the different classes. For this purpose, WLM uses class assignment rules and per-class resource shares and limits set by the system administrator. The resource entitlements and limits are enforced at the class level. This is a way of defining classes of service and regulates the resource utilization of each class of applications to prevent applications with very different resource utilization patterns from interfering with each other when they are sharing a single server.

Hierarchy of classes

WLM allows system administrators to set up a hierarchy of classes with two levels by defining superclasses and subclasses. In other words, a class can

either be a superclass or a subclass. The main difference between superclasses and subclasses is the resource control (shares and limits):

- At the superclass level, the determination of resource entitlement (based on the resource shares and limits) is based on the total amount of each resource managed by WLM available on the machine.
- At the subclass level, the resource shares and limits are based on the amount of each resource allocated to the parent superclass.

The system administrator (the root user) can delegate the administration of the subclasses of each superclass to a *superclass administrator* (a non-root user), thus allocating a portion of the system resources to each superclass and then letting superclass administrators distribute the allocated resources among the users and applications they manage.

WLM supports 32 superclasses (27 user defined plus five predefined). In turn, each superclass can have 12 subclasses (10 user defined and two predefined, as shown in Figure 6-6). Depending on the needs of the organization, a system administrator can decide to use only superclasses or both superclasses and subclasses. An administrator can also use subclasses only for some of the superclasses.

Each class is given a name by the WLM administrator who creates it. A class name can be up to 16 characters long and can only contain uppercase and lowercase letters, numbers, and underscores (_). For a given WLM configuration, the names of all the superclasses must be different from one another, and the names of the subclasses of a given superclass must be different from one another. Subclasses of different superclasses can have the same name. The fully qualified name of a subclass is superclass_name.subclass_name.

In the remainder of this section, whenever the term *class* is used, it is applicable to both subclasses and superclasses. The following subsections describe both super- and subclasses in greater detail, as well as the backward compatibility WLM provides to configurations of its first release.

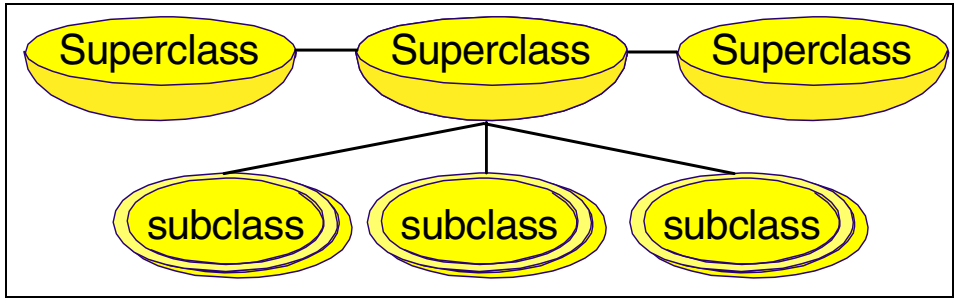


Figure 6-6 Hierarchy of classes

Superclasses

A superclass is a class with subclasses associated with it. No process can belong to the superclass without also belonging to a subclass, either predefined or user defined. A superclass has a set of class assignment rules that determines which processes will be assigned to it. A superclass also has a set of resource limitation values and resource target shares that determine the amount of resources that can be used by processes belonging to it. These resources will be divided among the subclasses based on the resource limitation values and resource target shares of the subclasses.

Up to 27 superclasses can be defined by the system administrator. In addition, five superclasses are automatically created to deal with processes, memory, and CPU allocation, as follows:

- *Default* superclass: The default superclass is named *default* and is always defined. All non-root processes that are not automatically assigned to a specific superclass will be assigned to the default superclass. Other processes can also be assigned to the default superclass by providing specific assignment rules.
- *System* superclass: This superclass has all privileged (root) processes assigned to it if they are not assigned by rules to a specific class, plus the pages belonging to all system memory segments, kernel processes, and kernel threads. Other processes can also be assigned to the system superclass. This default is for this superclass to have a memory minimum limit of one percent.
- *Shared* superclass: This superclass receives all the memory pages that are shared by processes in more than one superclass. This includes pages in shared memory regions and pages in files that are used by processes in more than one superclass (or in subclasses of different superclasses). Shared memory and files used by multiple processes that belong to a single superclass (or subclasses of the same superclass) are associated with that superclass. The pages are placed in the shared superclass only when a process from a different superclass accesses the shared memory region or file. This superclass can have only physical memory shares and limits applied to it. It cannot have shares or limits for the other resource types, subclasses, or assignment rules specified. Whether a memory segment shared by the processes in the different superclasses is classified into the shared superclass, or remains in the superclass it was initially classified into depends on the value of the localshm attribute of the superclass the segment was initially classified into.

- *Unclassified* superclass: The processes in existence at the time WLM is started are classified according to the assignment rules of the WLM configuration being loaded. During this initial classification, all the memory pages attached to each process are charged either to the superclass the process belongs to (when not shared, or shared by processes in the same superclass) or to the shared superclass, when shared by processes in different superclasses. However, there are a few pages that cannot be directly tied to any processes (and thus to any class) at the time of this classification, and this memory is charged to the unclassified superclass; for example, pages from a file that has been closed. The file pages will remain in memory, but no process *owns* these pages; therefore, they cannot be charged to a specific class. Most of this memory will end up being correctly reclassified over time, when it is either accessed by a process, or freed and reallocated to a process after WLM is started. There are a few kernel processes, such as wait or lrud, in the unclassified superclass. Even though this superclass can have physical memory shares and limits applied to it, WLM commands do not allow you to set shares and limits or specify subclasses or assignment rules on this superclass.
- *Unmanaged* superclass: A special superclass named unmanaged will always be defined. No processes will be assigned to this class. This class will be used to accumulate the memory usage for all pinned pages in the system that are not managed by WLM. The CPU utilization for the waitprocs is not accumulated in any class. This is deliberate; otherwise, the system would always seem to be at 100 percent CPU utilization, which could be misleading for users when looking at the WLM or system statistics. This superclass can not have shares or limits for any other resource types, subclasses, or assignment rules specified.

Subclasses

A subclass is a class associated with exactly one superclass. Every process in the subclass is also a member of the superclass. Subclasses only have access to resources that are available to the superclass. A subclass has a set of class assignment rules that determine which of the processes assigned to the superclass will belong to it. A subclass also has a set of resource limitation values and resource target shares that determine the resources that can be used by processes in the subclass. These resource limitation values and resource target shares indicate how much of the superclass's target (the resources available to the superclass) can be used by processes in the subclass.

Up to 10 out of a total of 12 subclasses can be defined by the system administrator or by the superclass administrator for each superclass. In addition, two special subclasses, default and shared, are always defined in each superclass as follows:

- *Default* subclass: The default subclass is named *default* and is always defined. All processes that are not automatically assigned to a specific subclass of the superclass will be assigned to the default subclass. You can also assign other processes to the default subclass by providing specific assignment rules.
- *Shared* subclass: This subclass receives all the memory pages used by processes in more than one subclass of the superclass. This includes pages in shared memory regions and pages in files that are used by processes in more than one subclass of the same superclass. Shared memory and files used by multiple processes that belong to a single subclass are associated with that subclass. The pages are placed in the shared subclass of the superclass only when a process from a different subclass of the same superclass accesses the shared memory region or file. There are no processes in the shared subclass. This subclass can only have physical memory shares and limits applied to it. It cannot have shares or limits for the other resource types or assignment rules specified.

Tiers

Tier configuration is based on the importance of a class relative to other classes in WLM. There are 10 available tiers from 0 through to 9. Tier value 0 is the most important and value 9 is the least important. As a result, classes belonging to tier 0 will get resource allocation priority over classes in tier 1, classes in tier 1 will have priority over classes in tier 2, and so on. The default tier number, if the attribute is not specified, is 0.

The tier applies at both the superclass and subclass levels. Superclass tiers are used to specify resource allocation priority between superclasses, and subclass tiers are used to specify resource allocation priority between subclasses of the same superclass. There is no relationship between tier numbers of subclasses of different superclasses.

Tier separation, in terms of prioritization, is much more enforced in AIX 5L than in the previous release. A process in tier 1 will never have priority over a process in tier 0, since there is no overlapping of priorities in tiers. It is unlikely that classes in tier 1 will acquire any resources if the processes in tier 0 are consuming all the resources. This occurs because the control of leftover resources is much more restricted than in the AIX Version 4.3.3 release of WLM, as shown in Figure 6-7 on page 234.

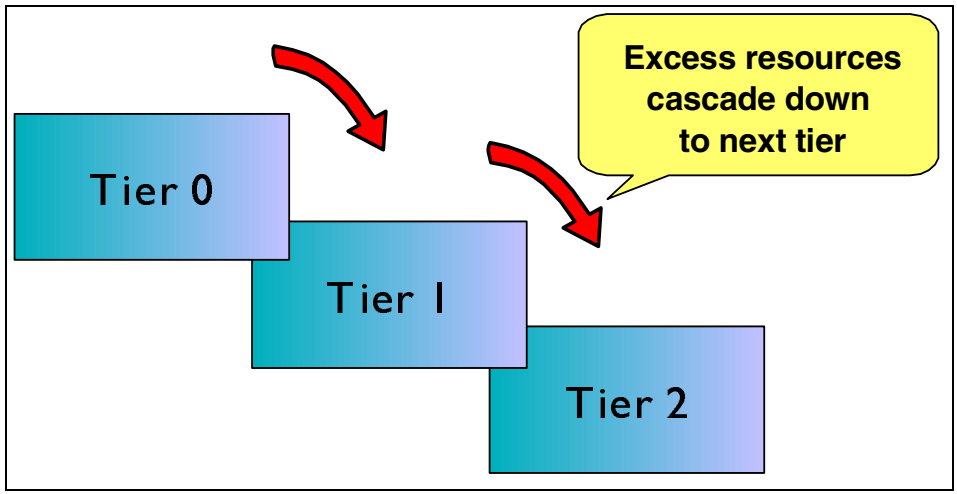


Figure 6-7 Resources cascading through tiers

Class attributes

In order to create a class, there are different attributes that are needed to have an accurate and well-organized group of classes. Figure 6-8 shows the SMIT panel for class attributes.

```
                        General characteristics of a class

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Class name                                          []
  Description                                         []
  Tier                                                [0]                    +#
  Resource Set                                                               +
  Inheritance                                         [No]                   +
  User authorized to assign its processes to this cl  []                     +
  ass
  Group authorized to assign its processes to this c  []                     +
  lass
  User authorized to administrate this class          []                     +
  (Superclass only)
  Group authorized to administrate this class         []                     +
  (Superclass only)


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 6-8 SMIT with the class creation attributes screen

The sequence of attributes within a class (as shown in Figure 6-8 on page 234) is outlined below:

- Class name

  A unique class name with up to 16 characters. It can contain uppercase and lowercase letters, numbers, and underscores (_).

- Description

  An optional brief description about this class.

- Tier

  A number between 0 and 9, for class priority ranking. It will be the tier that this class will belong to. An explanation about tiers can be found in “Tiers” on page 233.

- Resource set

  This attribute is used to limit the set of resources a given class has access to in terms of CPUs (processor set). The default, if unspecified, is *system*, which gives access to all the CPU resources available on the system.

- Inheritance

  The inheritance attribute indicates whether a child process should inherit its parent’s class or get classified according to the automatic assignment rules upon exec. The possible values are *yes* or *no*; the default is *no*. This attribute can be specified at both superclass and subclass level.

- User and group authorized to assign its processes to this class

  These attributes are valid for all the classes. They are used to specify the user name and the group name of the user or group authorized to manually assign processes to the class. When manually assigning a process (or a group of processes) to a superclass, the assignment rules for the superclass are used to determine which subclass of the superclass each process will be assigned to.

- User and group authorized to administer this class

  These attributes are valid only for superclasses. They are used to delegate the superclass administration to a user and group of users.

- Localchm

  Specifies whether memory segments that are accessed by processes in different classes remain local to the class they were initially assigned to, or if they go to the shared class.

Segment authorization to migrate to the shared class

With Workload Manager in earlier versions of AIX, whenever a memory segment is accessed by processes from different classes, the segment is reclassified as

shared. This occurs because one of the classes sharing the memory segment would otherwise be penalized as the user of this resource while the others are not. The consequence of the segment moving to shared is that users partially lose control of it. In AIX 5L, an attribute has been added at the class level to avert the automatic reclassification of the class. This attribute, localshm, if set to no, allows the segment to be reclassified to the shared class. If it is set to yes, then it is not reclassified. From the command line, the command will be similar to that shown in the example below:

```
# mkclass -a tier=2 -a adminuser=wlmu6 -a localshm=yes -c shares=2\
-m shares=3 -d new_config super3
```

From the SMIT panels, general characteristics of a class panel will have the localshm option, as in the example shown in Figure 6-9.

```
                    General characteristics of a class

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
  Class name                                         super3
  Description                                       []
  Tier                                              [2]                    +#
  Resource Set                                                             +
  Inheritance                                       [No]                   +
  User authorized to assign its processes to this cl []                    +
  ass
  Group authorized to assign its processes to this c []                    +
  lass
  User authorized to administrate this class        [wlmu6]                +
  (Superclass only)
  Group authorized to administrate this class       []                     +
  (Superclass only)
  Localshm                                          [Yes]                  +

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Figure 6-9 SMIT panel shows the additional localshm attribute

Classification process

There are two ways to classify processes in WLM:

- Automatic assignment when a process calls the system call exec, using assignment rules specified by a WLM administrator. This automatic assignment is always in effect (cannot be turned off) when WLM is active. This is the most common method of assigning processes to the different classes.

- Manual assignment of a selected process or group of processes to a class by a user with the required authority on both the process and the target class. This manual assignment can be done either by a WLM command, which could be invoked directly or through SMIT or Web-based System Manager, or by an application, using a function of the WLM Application Programming Interface. Manual assignment overrides automatic assignment.

6.6.2 Automatic assignment

The automatic assignment of processes to classes uses a set of class assignment rules specified by a WLM administrator. There are two levels of assignment rules:

- A set of assignment rules at the WLM configuration level used to determine which superclass a given process should be assigned to.
- A set of assignment rules at the superclass level used to determine which subclass of the superclass the process should be assigned to.

The assignment rules at both levels have exactly the same format.

When a process is created by fork, it remains in the same class as its parent. Usually, reclassification happens when the new process calls the system call exec. In order to classify the process, WLM starts by examining the top level rules list for the active configuration to find out which superclass the process should belong to. For this purpose, WLM takes the rules one at a time, in the order they appear in the file, and checks the current values for the process attributes against the values and lists of values specified in the rule. When a match is found, the process will be assigned to the superclass named in the first field of the rule. Then the rules list for the superclass is examined in the same way to determine which subclass of the superclass the process should be assigned to. For a process to match one of the rules, each of its attributes must match the corresponding field in the rule. The rules to determine whether the value of a process attribute matches the values in the field of the rules list are as follows:

- If the field in the rule has a value of hyphen (-), any value of the corresponding process attribute is a match.
- If the value of the process attribute (for all the attributes except *type)* matches one of the values in the list in a rule, and it is not excluded (prefaced by an exclamation point (!)), it is considered a match.
- When one of the values for the type attribute in the rule is comprised of two or more values separated by a plus (+) sign, a process will be a match for this value only if its characteristics match all the values mentioned above.

As previously mentioned, at both superclass and subclass levels, WLM goes through the rules in the order in which they appear in the rules list, and classifies the process in the class corresponding to the first rule for which the process is a match. This means that the order of the rules in the rules list is extremely important, and caution must be applied when modifying it in any way.

6.6.3 Manual assignment

Manual assignment is a feature introduced in AIX 5L WLM. It allows system administrators and applications to override, at any time, the traditional WLM automatic assignment (processes' automatic classification based on class assignment rules) and force a process to be classified in a specific class.

The manual assignment can be made or canceled separately at the superclass level, the subclass level, or both. In order to manually assign processes to a class or cancel an existing manual assignment, a user must have the right level of privilege (that is, must be the root user, adminuser, or admingroup for the superclass or authuser and authgroup for the superclass or subclass). A process can be manually assigned to a superclass only, a subclass only, or to a superclass and a subclass of the superclass. In the latter case, the dual assignment can be done simultaneously (with a single command or API call) or at different times, possibly by different users.

A manual assignment will remain in effect (and a process will remain in its manually assigned class) until:

- The process terminates.
- WLM is stopped. When WLM is restarted, the manual assignments in effect when WLM was stopped are lost.
- The class the process has been assigned to is deleted.
- A new manual assignment overrides a prior one.
- The manual assignment for the process is canceled.

In order to assign a process to a class or cancel a prior manual assignment, the user must have authority both on the process and on the target class. These constraints translate into the following:

- The root user can assign any process to any class.
- A user with administration privileges on the subclasses of a given superclass (that is, the user or group name matches the attributes adminuser or admingroup of the superclass) can manually reassign any process from one of the subclasses of this superclass to another subclass of the superclass.

- A user can manually assign their own processes (same real or effective user ID) to a superclass or a subclass for which they have manual assignment privileges (that is, the user or group name matches the attributes authuser or authgroup of the superclass or subclass).

This defines three levels of privilege among the persons who can manually assign processes to classes, root being the highest. In order for a user to modify or cancel a manual assignment, the user must be at the same or higher level of privilege as the person who issued the last manual assignment.

Class assignment rules

After the definition of a class, it is time to set up the class assignment rules so that WLM can perform its automatic assignment. The assignment rules are used by WLM to assign a process to a class based on the user, group, application path name, type of process, and application tag, or a combination of these five attributes.

The next sections describe the attributes that constitute a class assignment rule. All these attributes can contain a hyphen, which means that this field will not be considered when assigning classes to a process.

Class name

This field must contain the name of a class that is defined in the class file corresponding to the level of the rules file we are configuring (either superclass or subclass). Class names can contain only uppercase and lowercase letters, numbers, and underscores (_), and can be up to 16 characters in length. No assignment rule can be specified for the system-defined classes *unclassified*, *unmanaged,* and *shared*.

Reserved

Reserved for future use. Its value *must* be a hyphen, and it must be present in the rule.

User

The user name (as specified in the /etc/passwd file, LDAP, or in NIS) of the user owning a process can be used to determine the class to which the process belongs. This attribute is a list of one or more user names, separated by a comma. Users can be excluded by using an exclamation point prefix. Patterns can be specified to match a set of user names using full Korn shell pattern matching syntax.

Applications that use the setuid permission to change the *effective* user ID they run under are still classified according to the user that invoked them. The processes are only reclassified if the change is done to the *real* user ID (UID).

Group

The group name (as specified in the /etc/group file, LDAP, or in NIS) of a process can be used to determine the class to which the process belongs. This attribute is a list composed of one or more groups, separated by a comma. Groups can be excluded by using an exclamation point prefix. Patterns can be specified to match a set of group names using full Korn shell pattern matching syntax.

Applications that use the **`setgid`** permission to change the effective group ID they run under are still classified according to the group that invoked them. The processes are only reclassified if the change is done to the real group ID (GID).

Application path names

The full path name of the application for a process can be used to determine the class to which a process belongs. This attribute is a list composed of one or more applications, separated by a comma. The application path names will be either full path names or Korn shell patterns that match path names. Application path names can be excluded by using an exclamation point prefix.

Process type

In AIX 5L, the process type attribute is introduced as one of the ways to determine the class to which a process belongs. This attribute consists of a comma-separated list, with one or more combinations of values, separated by a plus sign (+). A plus sign provides a Boolean *and* function, and a comma provides a logical *or* function. Table 6-5 provides a list of process types that can be used. (Note: *32bit* and *64bit* are mutually exclusive.)

Table 6-5 List of process types

Attribute value	Process type
32bit	The process is a 32-bit process.
64bit	The process is a 64-bit process.
plock	The process called plock() to pin memory.
fixed	The process has a fixed priority (SCHED_FIFO or SCHED_RR).

Application tags

In AIX 5L, the application tag attribute is introduced as one of the forms of determining the class to which a process belongs. This is an attribute meant to be set by WLM's API as a way to further extend the process classification possibilities. This process was created to allow differentiated classification for different instances of the same application. This attribute can have one or more application tags, separated by commas. An application tag is a string of up to 30 alphanumeric characters.

The classification is done by comparing the value of the attributes of the process at exec time against the lists of class assignment rules to determine which rule is a match for the current value of the process attributes. The class assignment is done by WLM:

- When WLM is started for all the processes existing at that time.
- Every time a process calls the system calls exec, setuid (and related calls), setgid (and related calls), setpri, and plock, once WLM is started.

There are two *default* rules that are always defined (that is, hardwired in WLM). These are the default rules that assign all processes started by the user root to the system class, and all other processes to the default class. If WLM does not find a match in the assignment rules list for a process, these two rules will be applied (the rule for system first), and the process will go to either system (UID root) or default. These default rules are the only assignment rules in the standard configuration installed with AIX.

Table 6-6 is an example of classes with their respective attributes for assignment rules.

Table 6-6 Examples of class assignment rules

Class	Reserved	User	Group	Application	Type	Tag
System	-	root	-	-	-	-
db1	-	-	-	/usr/oracle/bin/db*	-	_db1
db2	-	-	-	/usr/oracle/bin/db*	-	_db2
devlt	-	-	dev	-	32bit	-
VPs	-	bob,!ted	-	-	-	-
acctg	-	-	acct*	-	-	-

In Table 6-6 the rule for default class is omitted from display, though this class's rule is always present in the configuration. The rule for system is explicit, and has been put first in the file. This is deliberate so that all processes started by root will be assigned to the system superclass. By moving the rule for the system superclass further down in the rules file, the system administrator could have chosen to assign the root processes that would not be assigned to another class (because of the application executed, for example) to system only. In Table 6-6, with the rule for system on top, if root executes a program in /usr/oracle/bin/db* set, the process will be classified as system. If the rule for the system class were after the rule for the db2 class, the same process would be classified as db1 or db2, depending on the tag.

These examples show that the order of the rules in the assignment rules file is very important. The more specific assignment rules should appear first in the rules file, and the more general rules should appear last. An extreme example would be putting the default assignment rule for the default class, for which every process is a match, first in the rules file. That would cause every process to be assigned to the default class (the other rules would, in effect, be ignored).

You can define multiple assignment rules for any given class. You can also define your own specific assignment rules for the system or default classes. The default rules mentioned previously for these classes would still be applied to processes that would not be classified using any of the explicit rules.

6.6.4 Backward compatibility

As mentioned earlier, in the first release of WLM, the system default for the resource shares was one share. In AIX 5L, it is -, which means that the resource consumption of the class for this particular resource is not regulated by WLM. This changes the semantics quite a bit, and it is advisable that system administrators review their existing configurations and consider if the new default is good for their classes, or if they would be better off either setting up a default of one share (going back to the previous behavior) or setting explicit values for some of the classes.

In terms of limits, the first release of WLM only had one maximum, not two. This maximum limit was in fact a *soft* limit for CPU and a *hard* limit for memory. Limits specified for the old format, *min percent-max percent*, will have, in AIX 5L, the max interpreted as a softmax for CPU and both values of hardmax and softmax for memory. All interfaces (SMIT, AIX commands, and Web-based System Manager) will convert all data existing from its old format to the new one.

The disk I/O resource is new for the current version, so when activating the AIX 5L WLM with the configuration files of the first WLM release, the values for the shares and the limits will be the default ones for this resource. The system defaults are:

- shares = -
- min = 0 percent, softmax = 100 percent, hardmax = 100 percent

For existing WLM configurations, the disk I/O resource will not be regulated by WLM, which should lead to the same behavior for the class as with the first version.

6.6.5 Resource sets

WLM uses the concept of resource sets (or rsets) to restrict the processes in a given class to a subset of the system's physical resources. In AIX 5L, the physical resources managed are the memory and the processors. A valid resource set is composed of memory and at least one processor.

Figure 6-10 shows the SMIT panel where a resource set can be specified for a specific class.

```
                        General characteristics of a class

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Class name                                          Redbook
  Description                                        [Redbook example]
  Tier                                               [0]                          +#
  Resource Set                                        sys/cpu.00003               +
  Inheritance                                        [Yes]                        +
  User authorized to assign its processes to this cl [user_s]                     +
  ass
  Group authorized to assign its processes to this c [system]                     +
  lass
  User authorized to administrate this class         [user_s]                     +
  (Superclass only)
  Group authorized to administrate this class        [system]                     +
  (Superclass only)


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 6-10 Resource set definition for a specific class

By default, the system creates one resource set for all physical memory, one for all CPUs, and one separate set for each individual CPU in the system. The **lsrset** command lists all resource sets defined. A sample output for the **lsrset** command follows:

```
# lsrset -av
T  Name               Owner   Group   Mode    CPU  Memory  Resources
r  sys/sys0           root    system  r-----    4     511  sys/sys0
sys/node.00000 sys/mem.00000 sys/cpu.00003 sys/cpu.00002 sys/cpu.00001
sys/cpu.00000
r  sys/node.00000     root    system  r-----    4     511  sys/sys0
sys/node.00000 sys/mem.00000 sys/cpu.00003 sys/cpu.00002 sys/cpu.00001
sys/cpu.00000
r  sys/mem.00000      root    system  r-----    0     511  sys/mem.00000
r  sys/cpu.00003      root    system  r-----    1       0  sys/cpu.00003
r  sys/cpu.00002      root    system  r-----    1       0  sys/cpu.00002
```

```
r  sys/cpu.00001      root    system  r-----    1       0  sys/cpu.00001
r  sys/cpu.00000      root    system  r-----    1       0  sys/cpu.00000
```

6.6.6 Rset registry

As mentioned previously, some resource sets in AIX 5L are created, by default, for memory and CPU. It is possible to create different resource sets by grouping two or more resource sets and storing the definition in the rset registry.

The rset registry services enable system administrators to define and name resource sets so that they can then be used by other users or applications. In order to alleviate the risks of name collisions, the registry supports a two-level naming scheme. The name of a resource set takes the form name_space/rset_name. Both the namespace and rset_name may each be 255 characters in size, are case-sensitive, and may contain only upper and lower case letters, numbers, underscores, and periods. The namespace of sys is reserved by the operating system and used for rset definitions that represent the resources of the system.

The **SMIT rset** command has options to list, remove, or show a specific resource set used by a process and the management tools, as shown in Figure 6-11.

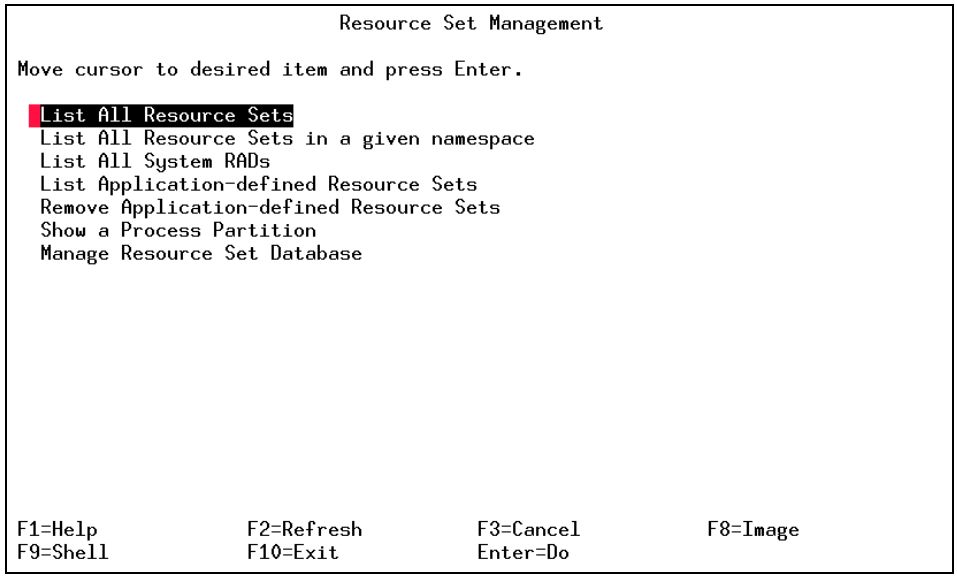


Figure 6-11 SMIT main panel for resource set management

To create, delete, or change a resource set in the rset registry, you must select the **Manage Resource Set Database** item in the SMIT panel. In this panel, it is

also possible to reload the rset registry definitions to make all changes available to the system. Figure 6-12 shows the SMIT panel for rset registry management.

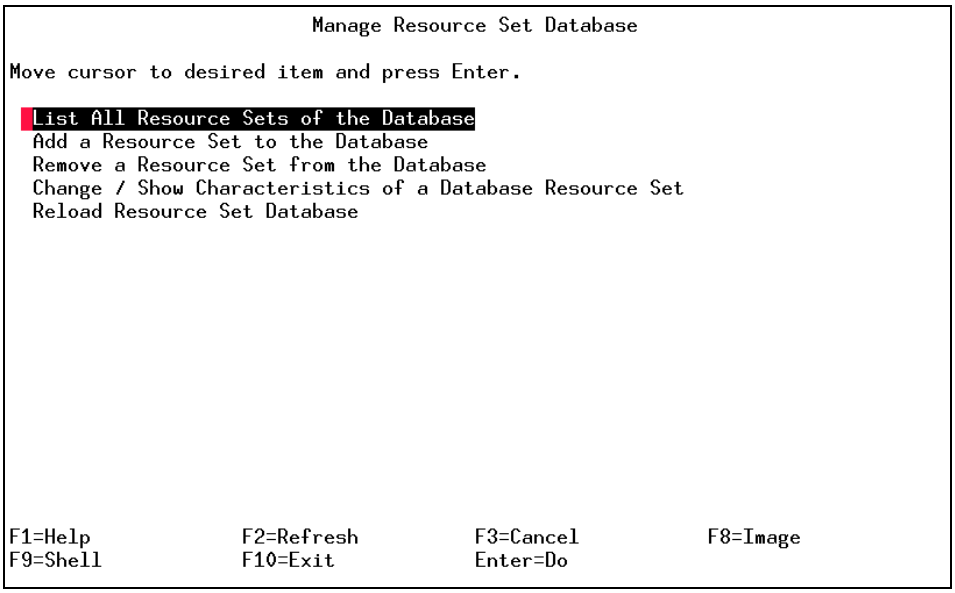


Figure 6-12 SMIT panel for rset registry management

To add a new resource set, you must specify a name space, a resource set name, and the list of resources. It is also possible to change the permissions for the owner and group of this rset. In addition, permissions for the owner, groups, and others can also be specified. Figure 6-13 on page 246 shows the SMIT panel for this task.

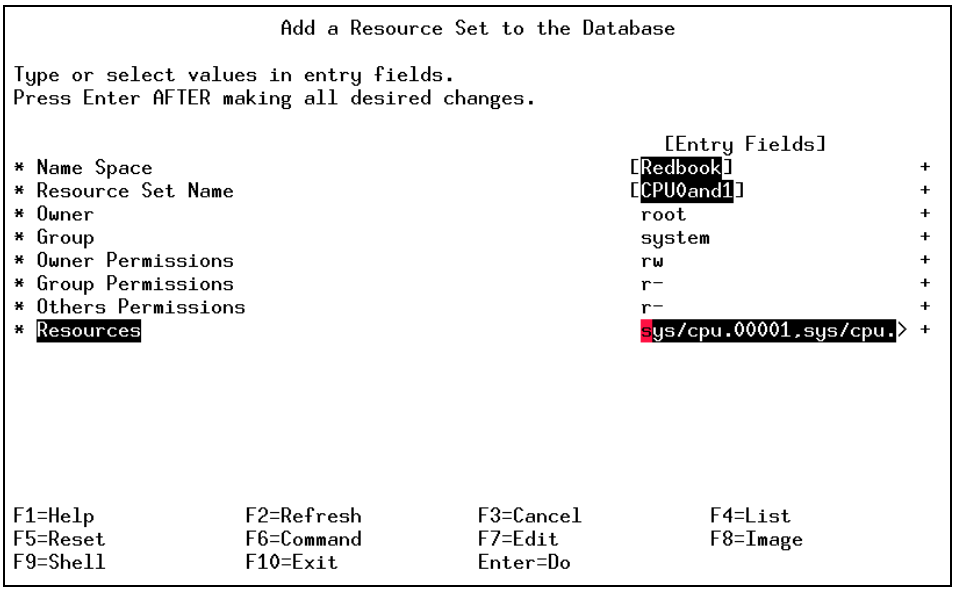


Figure 6-13 SMIT panel to add a new resource set

Whenever a new rset is created, deleted, or modified, a reload in the rset database is needed in order to make the changes effective.

After a WLM configuration has been defined by the system administrator, it can be made the active configuration using the **wlmcntrl** command. For example, to starts WLM in active mode, enter:

```
# wlmcntrl -a
```

To validate if WLM is running, enter:

```
# wlmcntrl -q
1495-052 WLM is running
```

To stop WLM, enter:

```
# wlmcntrl -o
```

6.7 Quiz

The following assessment questions help verify your understanding of the topics discussed in this chapter.

1. Which of the following should be implemented to balance available CPU/memory resources between applications?

   A. Loadleveler

   B. Nice/renice

   C. Resource Manager

   D. Workload Manager

2. A nice value for a process is shown as 80 in a **ps** listing. Which command is used to change this to a value of 100?

   A. **renice 100 PID**

   B. **renice -n 20 PID**

   C. **renice -20 PID**

   D. **renice -n -100 PID**

3. Which command changes operational parameters of the Virtual Memory Manager (VMM)?

   A. **nice**

   B. **renice**

   C. **schedtune**

   D. **vmtune**

4. Which of the following options is true regarding schedtune memory-load control?

   A. Memory-load control is intended to smooth out infrequent peaks in load that might otherwise cause a system to thrash. It is not intended to act continuously in a configuration that has too little RAM to handle its normal workload.

   B. Memory-load control cannot be turned off. Other parameters can be used to control thrashing, like **schedtune -D**.

   C. The AIX scheduler performs memory-load control by suspending processes when memory is overcommitted. The system then swaps out processes.

   D. Memory is considered overcommitted when the number of pages written to paging space in the last second, multiplied by the value of the -h parameter, is less than the number of page steals in the last second.

5. Which of the following WLM files do *not* contain data directly related to performance tuning?

   A. /etc/wlm/current/classes

   B. /etc/wlm/current/rules

   C. /etc/wlm/current/limits

   D. /etc/wlm/current/shares

6.7.1 Answers

The following are the preferred answers to the questions provided in this section.

1. D
2. B
3. D
4. A
5. B

6.8 Exercise

The following exercises provide sample topics for self study. They will help ensure comprehension of this chapter.

1. Find a process with a recent high CPU usage value. Use the **renice** command to lessen its priority value. Follow the process CPU utilization. Restore the nice value.
2. On a test system, manipulate the SCHED_R value to prioritize foreground processes (for reference, see Figure 6-4 on page 216). Restore the default values.

7

Performance scenario walkthroughs

This chapter provides a set of scenarios that allow you to better understand the relationship between the tools, their output, and a problem you may need to solve.

7.1 CPU performance scenario

In this section, a basic CPU-bound performance problem scenario is shown, with conclusions made based on output from commands previously discussed in this publication.

7.1.1 Data collection

The environment consists of a 2-way F50 with 50 Netstation clients connected over Ethernet. Users are using an HTML application as an interface to a database. Now the users are complaining about long response times. When starting a browser window on a Netstation, the start up seems slow. To verify this, the browser startup is executed with the **time** command:

```
# time netscape

real    0m16.73s
user    0m0.83s
sys     0m0.63s
```

By running **time netscape**, you can verify that the start up was slow. The normal start up time of a browser in the example system would be under 10 seconds. From the output, it seems that the systems waits for more than 15 seconds (user + sys = 1.46 seconds out of 16.73 seconds total time). In most cases systems wait for I/O, so you run **iostat**:

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        328.5              100.0      0.0        0.0       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        332.1              100.0      0.0        0.0       0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0
```

There is no activity against the disks, but the %user shows 100.0. (This is an extreme manufactured, although not edited, example). The problem is probably CPU related. Next you would likely check the run queue with the **vmstat** command:

```
# vmstat 2 5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 17354 15755   0   0   0   0    0   0 101   10   7 63  0 37  0
 5  1 17354 15754   0   0   0   0    0   0 407 2228 101 99  0  0  0
 5  1 17354 15752   0   0   0   0    0   0 413   43  93 99  0  0  0
 5  1 17354 15752   0   0   0   0    0   0 405   43  92 99  0  0  0
 5  1 17354 15752   0   0   0   0    0   0 407   42  90 99  0  0  0
```

7.1.2 Data analysis

Five jobs on the run queue is not a normal state for this system. The next step would be to find out what processes are causing the problems. This can be done with the **ps** command:

```
# ps -ef |sort +3 -r |head -15
    UID   PID  PPID   C    STIME    TTY  TIME CMD
thomasc 15860 12948  93 10:30:49  pts/1 17:41 ./tcprg5
thomasc 16312 12948  93 10:30:39  pts/1 20:30 ./tcprg3
thomasc 15234 12948  92 10:31:13  pts/1 15:21 ./tcprg1
thomasc 16844 12948  87 10:31:00  pts/1 13:15 ./tcprg2
thomasc 17420 12948  31 10:30:26  pts/1 14:53 ./tcprg4
    root 14778  3420   4 10:51:10  pts/3  0:00 ps -ef
    root 17154  3420   1 10:51:10  pts/3  0:00 sort +3 -r
    root 13676 15080   0 15:54:12  pts/5  0:00 ksh
    root 15080     1   0 15:54:11      -  0:00 xterm
    root  4980     1   0 15:37:42      -  0:00 /usr/lib/errdemon -s 2000000
    root 16510  3420   0 10:51:10  pts/3  0:00 head -15
    root 16022 10872   0   Jun 29   lft0  7:05 topas n
    root  3420  5568   0   Jun 28  pts/3  0:00 ksh
    root 12948 12796   0   Jun 28  pts/1  0:02 ksh
# ps auxwww |head -14
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME  TIME COMMAND
thomasc  16312 25.0 0.0   44   64  pts/1 A    10:30:39 26:28 ./tcprg3
root       516 24.0 3.0  264 15396      - A      Jun 28 9544:43 kproc
thomasc  15860 20.7 0.0   44   64  pts/1 A    10:30:49 21:49 ./tcprg5
thomasc  15234 20.6 0.0   44   60  pts/1 A    10:31:13 21:20 ./tcprg1
thomasc  16844 18.4 0.0   44   64  pts/1 A    10:31:00 19:13 ./tcprg2
thomasc  17420 15.7 0.0   44   64  pts/1 A    10:30:26 16:44 ./tcprg4
root      1032  6.7  3.0  264 15396      - A      Jun 28 2679:27 kproc
root      1290  3.2  3.0  264 15396      - A      Jun 28 1263:12 kproc
root       774  3.2  3.0  264 15396      - A      Jun 28 1258:58 kproc
root      3158  0.0  0.0  356  384      - A Jun 28 8:27/usr/sbin/syncd 60
```

```
root     16022  0.0  0.0  488  640   lftO A      Jun 29  7:05 topas n
root      2064  0.0  3.0  320 15452      - A      Jun 28  2:38 kproc
root         0  0.0  3.0  268 15400      - A      Jun 28  1:26 swapper
```

One user, thomasc, has started five programs with the prefix `tcprg` that has accumulated a lot of recent CPU usage (C column). When looking at the -u flag output from the **ps** command, the %CPU (reporting how much a process has used CPU since started), these test programs use abnormally high CPU.

7.1.3 Recommendation

There are several ways to reverse the overload (**kill PID**, for example), but the proper thing would be to check with the user thomasc and ask some questions, such as “What are these process?” “Why are they running. Can they be stopped?” “Do they have to run now. Can they be rescheduled?” Rescheduling these kind of CPU consuming processes to a less busy time will probably give the most significant advantage in performance.

If these processes can be rescheduled, this can be done with the **batch** command, the **at** command, or by starting them from the **crontab**. Remember to start such jobs at times when they are not in conflict with OLTPs.

If they have to run during a busy time, and to be running at times in the future, some changes need to be done to improve the balance the performance of the system. A recommendation is to move these test programs to a test system, excluded from the production environment.

Another solution is to buy more CPUs, which is a good step if the hardware can accommodate this, but may move the bottleneck to another resource of the system, for example, memory.

Resource capping could be a solution, and for that there is no better way than Workload Manager (WLM). For more information on WLM, see *AIX 5L Workload Manager (WLM)*, SG24-5977.

7.2 I/O performance scenario

In this scenario, a user reports that the month-end report is taking a long time to run and the user is unsure what is causing this. One possible reason for this report taking so long is that when AIX creates a print job the job is first written to the print spooler. This spool file is created on the disk in /var/adm/spool. If there is an I/O problem where the system is waiting for disk, then this file can take a long time to generate, especially if it is a large file.

7.2.1 Data collection

In this section, the outputs of the system are gathered by the **vmstat** and **iostat** commands.

Below is the output of the **iostat** command for this system.

```
# iostat 1 10

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.9         52.6                2.7     20.1       43.3      33.9

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          19.4     350.9      32.3     870967    921096
hdisk1          49.0     616.6      52.9    1267281   1881244
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          1.0          0.0                0.0     12.0        0.0      88.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          29.0    1616.0     101.0          0      1616
hdisk1         100.0    2164.0     108.0       1656       508
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          1.0         58.0                0.0      6.0        0.0      94.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          25.0     660.0      50.0          0       660
hdisk1         100.0    1108.0     111.0        672       436
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          2.0         58.0                0.0      6.0        0.0      94.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          18.0     208.0      21.0          4       204
hdisk1         100.0    1552.0     114.0        316      1236
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          2.0         94.0                0.0     12.0        0.0      88.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          18.0     232.0      28.0          0       232
hdisk1          98.0     808.0     111.0        312       496
cd0              0.0       0.0       0.0          0         0
```

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          1.0         47.0               0.0      6.0        0.0      94.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          12.0      80.0      20.0          4        76
hdisk1         100.0     804.0     105.0        188       616
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          2.0         94.0               0.0      9.0        0.0      91.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          17.0     216.0      21.0          0       216
hdisk1         100.0     916.0     103.0        328       588
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          2.0         48.0               0.0     13.0        0.0      87.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          18.0     184.0      19.0          0       184
hdisk1          99.0    1728.0     120.0        244      1484
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          1.0          1.0               0.0     20.8        0.0      79.2

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           8.9      67.3      13.9          4        64
hdisk1         100.0    3655.4     127.7        136      3556
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
         11.0         11.0               0.0      6.0        0.0      94.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          23.0     200.0      23.0          0       200
hdisk1         100.0     744.0     102.0        276       468
cd0              0.0       0.0       0.0          0         0
```

Below is the output of the system using the **vmstat** command:

```
# vmstat 1 10
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 19776   121   0   1  82  225  594   0 208  658 160  3 20 43 34
 0  2 19776   115   0   0   0  408  911   0 338 1160 327  0  9  0 91
 0  3 19776   121   0   0   0  410  950   0 329  971 300  0 12  0 88
```

```
 0  3 19776    121    0    0    0  337   724    0 335   950 360   0   9   0 91
 0  3 19776    120    0    0    0  562  1136    0 341  1279 256   0  19   0 81
 0  3 19776    119    0    0    0  632  1360    0 349  1230 247   1  11   0 88
 0  2 19776    118    0    0    0  641  1366    0 359  1630 281   0  19   0 81
 0  3 19776    121    0    0    0 1075  3353    0 362  2147 322   0  23   0 77
 0  3 19776    123    0    0    0  761  1700    0 367  1225 376   3  11   0 86
 0  3 19776    123    0    0    0 1170  1819    0 435  1374 390   0  21   0 79
```

7.2.2 Data analysis

In this section, the key indicators of the output will be looked at and an explanation given regarding the output.

The vmstat command output investigation

Although the **vmstat** command is a memory diagnostic tool, it does display one I/O column. Notice the cpu column with the wa output; the output is on average 85 percent (add the column, excluding the first line, and divide by nine). If the wa value is higher than 25 percent, it may indicate a problem with the disk subsystem.

The high wa value leads you to two additional columns. Under kthr, kernel threads, the b column indicates 2–3 threads per second are waiting. Under memory, the fre column, the number of buffer frames available on the free is very low.

The iostat command output investigation

The key values to check here are the % iowait and the % tm_act values. Remember that the first output is the current status since startup.

The % iowait value

The % iowait is the percentage of time the CPU is idle while waiting on local I/O.

In this example, the %iowait has an average of 89.9 percent (add the column, excluding the first line, and divide by nine). If the system's % iowait is higher than 25 percent, it is advisable to investigate the problem and take corrective action.

The % tm_act value

The %tm_act is the percentage of time the disk is busy.

In this example, the % tm_act value had an average of 18.8 percent for hdisk0 and an average of 99.7 percent for hdisk1. If the percentage for the time a disk is busy is high on a smaller system there will be noticeable performance degradation on the system with less disks. In order to deliver good performance, a system should be recording an average disk busy of less than 40 percent. This is, however, not always possible with smaller systems with less disks.

7.2.3 Recommendation

The following are some recommendations to assist in improving disk I/O:

- Look for idle drives on the system; it may be possible to move some data from busy drives to idle drives, which will give improved performance.
- Check the paging activity, as this may also be a factor. Spread the paging over multiple drives if possible, thus sharing the paging space load across multiple drives.
- If this is an occasional occurrence during month end, check which other I/O-intensive processes are running and if they can be run earlier or later; this way, the load is also spread across time.

7.3 Additional I/O scenarios

The following scenarios are examples of various command reports used as input for tuning a system that has an I/O performance problem.

7.3.1 CPU and kernel thread I/O wait bottleneck scenario

The following scenario provides the following **vmstat** command report as input:

```
$ /usr/bin/vmstat 120 10
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  1 59903   542   0   0   0   0    0   0 451  912 478 43 11 15 31
 0  2 59904   550   0   0   0   0    0   0 521 1436 650 23 19  4 50
 0  3 59950   538   0   0   0   0    0   0 344  649 249  7  7  6 80
 0  2 59899   578   0   0   0   0    0   0 467 1829 500 12 14  4 70
 0  2 59882   589   0   0   0   0    0   0 600 1292 705  6  8  3 61
 0  3 59882   420   0   0   0   0    0   0 452  952 372 11  8  1 80
 0  2 59954   420   0   0   0   0    0   0 537 1979 573 13  5 10 72
 0  2 59954   423   0   0   0   0    0   0 618 1413 686 15  9  6 70
 0  3 59954   420   0   0   0   0    0   0 551  938 634  4  2  2 92
 0  2 59954   422   0   0   0   0    0   0 460 1376 496 14  2  4 80
```

The **vmstat** report is taken over a period of 20 minutes using a 120 second interval repeated 10 times. The first figures that are interesting in this report are the cpu values (us/sy/id/wa). Notice that the CPU does have some idle (id) time, but the largest value is the I/O wait (wa) time. Remember that the first measurement can be excluded, because it is the average on the system since startup.

The I/O wait time is increasing over the sample period from 50 percent and peaking at 92 percent on the ninth measurement. The average I/O wait time over the sample period is 72 percent, which indicates an I/O-related bottleneck.

The wa column specifies the percentage of time the CPU was idle with pending local disk I/O. If there is at least one outstanding I/O to a local disk when wait is running, the time is classified as waiting for I/O.

Generally, a wa value over 25 percent indicates that the disk subsystem may not be balanced properly, or it may be the result of a disk-intensive workload.

Check the b value in the kernel thread column. The b column lists the average number of kernel threads placed on the wait queue per second and should stay near zero. These threads are waiting for resources or I/O.

Notice that the memory in relation to paging I/O can be ignored in this scenario, as the paging parameters are all zero and the list of free memory pages (fre) is still acceptable.

For more information on where the possible I/O bottleneck occurs, an **iostat** command should be run on this system. This will provide information about how the disk I/O is distributed between the physical volumes and where the possible bottlenecks could be.

7.3.2 I/O distribution bottleneck scenario

This scenario returned the following **lsps** and **iostat** report:

```
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
hd6         hdisk0            rootvg         256MB      13     yes   yes    lv
# iostat 120 5
...
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
         47.8       1394.6              50.3     19.6       25.0       5.1


Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          97.0     124.4      59.3       1924     12240
hdisk1           0.8      21.5      16.8        492         0
hdisk2           0.2       0.3       0.1          8        12

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
         47.1       1046.3              45.0     40.0        4.0      11.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          98.5     186.1      56.4       9260    13008
hdisk1           0.6      23.8      18.5         96      332
```

```
hdisk2          0.3       0.6       0.1          4       32

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          39.8       1709.1              30.0     40.0       10.0      20.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          98.3     164.6      55.2       7144    12532
hdisk1           0.2      36.9      26.6        312      904
hdisk2           1.2       2.3       0.5         36      100


tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          32.9       1467.4              30.6     37.4       22.0      10.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          99.8     183.9      22.6       1364    20576
hdisk1           0.6      35.6      16.8        672      464
hdisk2           0.5       1.2       0.3         24       48


tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          33.6        875.5              18.4     41.1       10.5      30.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          98.9     180.5       7.5       3132    18560
hdisk1           0.2      15.6       2.9         80      808
hdisk2           0.3       0.7       0.2         12       32
...
```

The **lsps** command shows that the paging space is allocated on the physical volume hdisk0.

The **iostat** report shown is collecting data over a 10-minute period in 120-second intervals. The first report is not shown and should be ignored for real-time analysis, as it is the historical disk I/O report since startup. For details on the **iostat** command report, see 4.2, “The iostat command” on page 118.

Notice the high I/O wait time (% iowait), which is increasing from 5.1 percent to 30 percent. Generally, if an I/O wait time exceeds 25 percent, there is a problem related to disk I/O. There might be cases where I/O wait time is 0 percent and there still is an I/O bottleneck. This can happen when the system is performing extensive paging and the swap device is overloaded.

In the above **iostat** report, the activity on the hdisk0 is extremely high. The % tm_act, indicating the active time of the disk in percent, is between 97 percent and 99.8 percent, which is almost constantly active. This indicates that the I/O bottleneck is bound to the disk hdisk0. Notice the Kb_wrtn value, indicating the data written to the disk is fairly high compared to the amount of data read from

the disk. This leads to the conclusion that the I/O limits of the hdisk0 have been reached. To improve the I/O performance, the other disks should be used more actively, by moving hot file, file system, and logical volumes to the less active disks. In general, a more insightful investigation may be required using other tools such as **filemon** and **lslv**.

7.3.3 Logical volume fragmentation scenario

The **lslv** command is used for displaying the attributes of logical volumes, such as the fragmentation of a logical volume on a physical volume.

Following is the **lspv** command output of a fragmented logical volume:

```
lslv -p hdisk0 lv00
hdisk0:lv00:N/A
0001   0002   0003   0004   0005   0006   0007   0008   0009   0010     1-10
0011   0012   0013   0014   0015   0016   0017   0018   0019   0020     11-20
0021   0022   0023   0024   0025   0026   0027   0028   0029   0030     21-30
0031   0032                                                           31-32

0033   0034   0035   0036   0037   0038   0039   0040   0041   0042     33-42
0043   0044   0045   0046   0047   0048   0049   0050   0051   0052     43-52
0053   0054   0055   0056   0057   0058   0059   0060   0061   0062     53-62
0063   0064                                                           63-64

USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     65-74
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     75-84
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     85-94
USED                                                                   95-95

USED   USED   USED   USED   USED   USED   USED   FREE   FREE   FREE     96-105
FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE   FREE     106-115
FREE   FREE   FREE   FREE   FREE   FREE   FREE   0065   0066   0067     116-125
0068   0069                                                           126-127

FREE   0070   0071   USED   USED   USED   USED   USED   USED   USED     128-137
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     138-147
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     148-157
USED   USED                                                           158-159
```

The logical volume consisting of 71 logical partitions (LPs) is fragmented over four of the five possible intra-disk allocation sections. The sections' outer edge and outer middle are allocated from LPs 01–32 and 33–64, respectively, on similar contiguously physical partitions. The LPs 65–69 are allocated in the inner middle section and the last two LPs are allocated in the inner edge section.

This obvious fragmentation of logical volume lv00 can lead to a decrease in I/O performance due to longer seek and disk head changes for a sequential

read/write operation in the last part of the logical volume. As there is free space left on the physical volume, reorganizing lv00 is the action to take. Using the **reorgvg** command on this logical volume will help improve the performance of lv00.

For more information on the **lslv** command refer to 4.4, “LVM performance analysis using the lslv command” on page 131.

7.3.4 Monitoring scenario using filemon

Consider a system with the following disks available:

```
# lspv
hdisk0          000bc6fdc3dc07a7    rootvg
hdisk1          000bc6fdbff75ee2    none
```

The following output shows a section of a **filemon** report made for monitoring the logical volumes of a system.

The report was generated with the **filemon** command **filemon -O lv -o filemon.out**:

```
...
Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume                   description
------------------------------------------------------------------------
  0.84 105792 149280   177.1   /dev/hd1                 /home
  0.32      0  16800    11.9   /dev/hd8                 jfslog
  0.03      0   4608     3.2   /dev/hd4                 /
  0.02    864  55296     5.9   /dev/hd2                 /usr
  0.02    192   4800     3.5   /dev/hd9var              /var
  0.01      0   2976     2.1   /dev/hd8                 jfslog
...
```

The output shows that the logical volume hd1, containing the /home file system, has by far the highest utilization. As the second physical volume hdisk1 is not used, as seen in the **lspv** report above, it would be possible to add this physical volume to the rootvg and distribute hd1 to use both hdisk0 and hdisk1. This can either be done by splitting the logical volume using an inter-disk policy of maximum or by using the striping option.

7.3.5 Logical volume allocation scenario

The following scenario shows a series of status commands from a system with an allocation problem on a dedicated database volume group:

```
# lsps -s
```

```
Total Paging Space   Percent Used
      100MB              37%

# lsps -a
Page Space  Physical Volume Volume Group    Size   %Used  Active  Auto  Type
hd6         hdisk0            rootvg         100MB      38     yes   yes    lv
```

This shows that the paging space is defined in the hdisk0 of the rootvg.

The following volume group information is provided:

```
# lsvg
rootvg
datavg

# lspv
hdisk0        000038744c632197    rootvg
hdisk1        00002199abf65a1a    datavg
hdisk2        00000228b9c5d7da    datavg
hdisk3        00002199b40b728c    datavg
```

This shows that two volume groups are defined, rootvg on hdisk1 and datavg on hdisk1, hdisk2, and hdisk3.

hdisk0 contains the following logical volumes:

```
# lspv -l hdisk0
hdisk0:
LV NAME               LPs   PPs   DISTRIBUTION          MOUNT POINT
hd5                   2     2     02..00..00..00..00    N/A
hd3                   6     6     02..00..04..00..00    /tmp
hd2                   117   117   00..47..42..28..00    /usr
hd8                   1     1     00..00..01..00..00    N/A
hd4                   2     2     00..00..02..00..00    /
hd9var                1     1     00..00..01..00..00    /var
hd6                   128   128   29..50..49..00..00    N/A
hd1                   3     3     00..00..01..02..00    /home
lv00                  10    10    00..00..00..10..00    /database
```

The rootvg on hdisk0 contains all the default logical volume and file systems and an additional lv00 containing the /database file system.

The other disk contains:

```
# lspv -l hdisk1
hdisk1:
LV NAME               LPs   PPs   DISTRIBUTION          MOUNT POINT
loglv00               1     1     01..00..00..00..00    N/A
lv01                  10    10    00..00..00..00..10    /db01
```

```
lv02                10    10    00..00..00..00..10    /db02
lv03                10    10    10..00..00..00..00    /db03

# lspv -l hdisk2
hdisk2:
LV NAME             LPs   PPs   DISTRIBUTION          MOUNT POINT

# lspv -l hdisk3
hdisk3:
LV NAME             LPs   PPs   DISTRIBUTION          MOUNT POINT
```

The logical volumes of the datavg volume group are all allocated on the same physical volume hdisk1, including the jfs log loglv00. This shows that the physical volumes hdisk2 and hdisk3 are unused.

The details of the datavg LVs are as follows:

```
# lslv lv01
LOGICAL VOLUME:     lv01                   VOLUME GROUP:   datavg
LV IDENTIFIER:      0000881962b29b51.1     PERMISSION:     read/write
VG STATE:           active/complete        LV STATE:       opened/syncd
TYPE:               jfs                    WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        8 megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                1                      PPs:            1
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    yes
INTRA-POLICY:       middle                 UPPER BOUND:    32
MOUNT POINT:        /db01                  LABEL:          /db01
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes

# lslv lv02
LOGICAL VOLUME:     lv02                   VOLUME GROUP:   datavg
LV IDENTIFIER:      0000881962b29b51.3     PERMISSION:     read/write
VG STATE:           active/complete        LV STATE:       opened/syncd
TYPE:               jfs                    WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        8 megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                1                      PPs:            1
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    yes
INTRA-POLICY:       middle                 UPPER BOUND:    32
MOUNT POINT:        /db02                  LABEL:          /db02
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes

# lslv lv03
LOGICAL VOLUME:     lv03                   VOLUME GROUP:   datavg
```

```
LV IDENTIFIER:      0000881962b29b51.4     PERMISSION:     read/write
VG STATE:           active/complete        LV STATE:       opened/syncd
TYPE:               jfs                    WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        8 megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                1                      PPs:            1
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    yes
INTRA-POLICY:       middle                 UPPER BOUND:    32
MOUNT POINT:        /db03                  LABEL:          /db03
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes
```

When generating the logical volumes lv01, lv02, and lv03, the system administrator should have dedicated each of the LVs on a corresponding physical volume. Alternatively, the inter-disk allocation policy could have been set to maximum and limit the upper bound to 1.

In this way, the lv01 and the corresponding /db01 would reside on hdisk1, lv02 and /db02 on hdisk2, and lv03 and /db03 on hdisk3.

A modification to this can be done using the **migratepv** command.

To distribute the load of the default jfslog on the hdisk1 to the other physical volume in the datavg, an additional jfslog for each file system could be created. Defining a dedicated JFS log for both /db02 and /db03 would improve the performance. In this way, the different file systems residing on their individual disks would not utilize the hdisk1 for the JFS logging, but rather their own physical volumes.

7.4 Paging performance scenario

In this section, paging performance will be investigated. The symptoms of high memory utilization will be described and possible corrective action will be explained.

7.4.1 Data collection

In the following section, the system outputs are gathered by the **svmon** and **vmstat** commands.

Below is the output of an idle system using the **vmstat** command:

```
# vmstat 1 5
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy   in   sy  cs us sy id wa
 0  0 11106 107916   0   0   0   0    0   0 125  570  66  1  4 88  7
 0  0 11106 107915   0   0   0   0    0   0 112  397  42  0  0 98  2
 0  0 11106 107915   0   0   0   0    0   0 107  192  23  0  0 99  0
 0  0 11106 107915   0   0   0   0    0   0 110  280  28  0  0 99  0
 0  0 11106 107915   0   0   0   0    0   0 109  174  27  1  0 99  0
```

Below is the output of a system with high memory utilization using the **vmstat** command:

```
# vmstat 1 15
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in     sy  cs us sy id wa
 0  0 204340    72   0   0   5   8   25   0 108    249  29  0  1 98  1
 3  4 204649   124   0  31 422 449  912   0 347   4796 350  5 95  0  0
 1  3 204988   112   0  56 183 464 1379   0 339  14144 382  4 96  0  0
 9  0 205292   122   0  24 251 369  988   0 352    598 403  4 94  0  2
 3  1 205732   119   0   0 409 520  771   0 313    780 293  1 99  0  0
 3  1 206078   123   0   0 445 496  602   0 336    706 298  2 98  0  0
 3  1 206460   120   0   0 343 504 1210   0 305    719 271  1 99  0  0
 2  1 206897   119   0   0 320 512  981   0 311    660 288  0 99  0  0
 3  1 207186   126   0   1 369 504  929   0 331    718 292  1 99  0  0
 3  1 207491   120   0   1 428 504  844   0 319    763 262  1 99  0  0
 4  0 207964   119   0   0 275 520  791   0 296    632 283  0 99  0  0
 4  0 208354   119   0   2 373 513  816   0 328    664 297  1 99  0  0
 4  0 208715    87   0   4 383 464  753   0 330   1480 261  4 96  0  0
 3  1 209006     4   0  12 282 504  630   0 350 1 385 286  2 98  0  0
 3  2 209307    10   0   0 316 488  685   0 320    635 287  1 92  0  7
```

The following command outputs will be used for reference purposes or as comparisons. Each of these outputs were taken during the **vmstat** output above.

The output of the **ps** command appears as follows:

```
# ps gv | head -n 1; ps gv | egrep -v "RSS" | sort +6b -7 -n -r
PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
12478  pts/4 A     2:05   91 742240 362552 32768     2     4 50.8 69.0 ./tmp/me
  1032      - A     0:56    0    64  6144    xx     0  6088  0.0  1.0 kproc
   774      - A     0:01    0    16  6104    xx     0  6088  0.0  1.0 kproc
  7484      - A     0:00    6    16  6104 32768     0  6088  0.0  1.0 kproc
 10580      - A     0:00    1    16  6104 32768     0  6088  0.0  1.0 kproc
     0      - A     0:20    7    12  6100    xx     0  6088  0.0  1.0 swapper
   516      - A    3920:23    0     8  6096    xx     0  6088 98.7  1.0 kproc
  2076      - A     0:00    0    16  6096    xx     0  6088  0.0  1.0 kproc
```

```
  3622      - A     0:00    0    16  6096    xx     0  6088  0.0  1.0 kproc
  7740      - A     0:00    0    16  6096 32768     0  6088  0.0  1.0 kproc
  4994  pts/5 A     0:00   24   440   708 32768   198   220  0.0  0.0 ksh /usr/
 15434  pts/5 A     0:00    0   368   396 32768   198   220  0.0  0.0 ksh /usr/
  4564  pts/0 A     0:00    0   308   392 32768   198   220  0.0  0.0 -ksh
 15808  pts/2 A     0:00  292   304   388 32768   198   220  0.0  0.0 ksh
  5686  pts/0 A     0:00  320   280   348 32768   198   220  0.0  0.0 -ksh
 11402  pts/1 A     0:00  225   296   336 32768   198   220  0.0  0.0 -ksh
  2622      - A     0:39  469  3208   324    xx  2170   112  0.0  0.0 /usr/lpp/
 16114  pts/0 A     0:00    0   240   324 32768    52    60  0.0  0.0 ps gv
 16236  pts/5 A     0:00   12   360   252 32768   198   220  0.0  0.0 ksh /usr/
 11982  pts/4 A     0:00  160   304   240 32768   198   220  0.0  0.0 -ksh
 13934  pts/2 A     0:00  234   304   236 32768   198   220  0.0  0.0 -ksh
 14462  pts/3 A     0:00  231   308   232 32768   198   220  0.0  0.0 -ksh
 16412  pts/5 A     0:00  129   304   232 32768   198   220  0.0  0.0 -ksh
1       - A     0:07  642   760   224 32768    25    36  0.0  0.0 /etc/init
  6708      - A     0:02  394   728   212 32768   337    80  0.0  0.0 /usr/sbin
  6212      - A     0:00  567   644   208 32768   327    64  0.0  0.0 sendmail:
  3124      - A     5:22  340  1152   204    xx    40     0  0.1  0.0 dtgreet
 17316  pts/0 A     0:00   71    88   196 32768    43    68  0.0  0.0 svmon -i
 17556  pts/0 A     0:00    1   148   196 32768    16    24  0.0  0.0 egrep -v
 12886  pts/3 A     1:53 30625   228   192 32768    10    12  9.8  0.0 cp -r
/u/
 16960  pts/0 A     0:00   40   132   184 32768    15    20  0.0  0.0 vmstat 1
 13104  pts/1 A     0:00   63   132   156 32768    15    20  0.0  0.0 vmstat 1
 13466  pts/5 A     0:00    0   104   136 32768     2     4  0.0  0.0 /usr/bin/
  4774      - A     0:00  217   284   124 32768    30    28  0.0  0.0 /usr/sbin
 13796  pts/5 A     0:00    4    80    76 32768    18    24  0.0  0.0 dd conv=s
 14856  pts/5 A     0:01    0    72    64 32768    18    24  1.1  0.0 dd conv=s
  5440      - A     0:00  228   292    60 32768    25    20  0.0  0.0 /usr/sbin
  9292      - A     0:00  183   128    60 32768    53    20  0.0  0.0 /usr/sbin
  1920      - A     0:50 16272    96    36    xx     2     4  0.0  0.0
/usr/sbin
 14198      - A     0:00  274   740    20 32768   313     4  0.0  0.0 telnetd -
  2516      - A     0:00    0   656    16 32768   313     4  0.0  0.0 telnetd -
  8000      - A     0:00   51   656    16 32768   313     4  0.0  0.0 telnetd -
  8780      - A     0:00   19   120    16 32768     3     0  0.0  0.0 /usr/sbin
 11614      - A     0:00    9   180    16 32768    18     0  0.0  0.0 /usr/lpp/
 12788      - A     0:00  102   740    16 32768   313     4  0.0  0.0 telnetd -
14710      - A     0:00  350   740    16 32768   313     4  0.0  0.0 telnetd -
 15298      - A     0:00    0   740    16 32768   313     4  0.0  0.0 telnetd -
  2874      - A     0:00   29   288    12    xx   100     0  0.0  0.0 /usr/dt/b
  3402      0 A     0:00    5   180    12    xx    34     0  0.0  0.0 slattach
  3900      - A     0:00  442   460    12    xx    56     0  0.0  0.0 /usr/lib/
  4134      - A     0:00   26   400    12    xx   100     0  0.0  0.0 dtlogin <
  5176      - A     0:00   44   456    12 32768    31     0  0.0  0.0 /usr/sbin
  5938      - A     0:00   37   280    12 32768    36     0  0.0  0.0 /usr/sbin
  6450      - A     0:00   99   304    12 32768    25     0  0.0  0.0 /usr/sbin
  6966      - A     0:00   52   428    12 32768   190     0  0.0  0.0 /usr/sbin
```

```
  7224      - A     0:00   56   500    12 32768   191     0  0.0  0.0 /usr/sbin
  8260      - A     0:00    1    96    12 32768     2     0  0.0  0.0 /usr/sbin
  8522      - A     0:00   13   292    12 32768    21     0  0.0  0.0 /usr/sbin
  9040      - A     0:00    3    36    12 32768     5     0  0.0  0.0 /usr/sbin
  9554      - A     0:00    5   220    12 32768    12     0  0.0  0.0 /usr/sbin
  9808      - A     0:00   12   312    12 32768    64     0  0.0  0.0 /usr/bin/
 10838   lft0 A     0:00   17   372    12 32768    40     0  0.0  0.0 /usr/sbin
 11094      - A     0:00   13   256    12 32768    22     0  0.0  0.0 /usr/IMNS
```

The output of the **svmon** command appears as follows:

```
# svmon -i 5 5
size      inuse       free        pin    virtual
memory       131063     130936        127       6946     204676
pg space     131072     106986

               work       pers       clnt
pin             6955          0          0
in use        104809      26127          0


               size      inuse       free        pin    virtual
memory       131063     130942        121       6942     206567
pg space     131072     108647

               work       pers       clnt
pin             6951          0          0
in use        105067      25875          0


               size      inuse       free        pin    virtual
memory       131063     130951        113       6942     208406
pg space     131072     110432

work       pers       clnt
pin             6955          0          0
in use        104809      26127          0


               size      inuse       free        pin    virtual
memory       131063     130942        121       6942     206567
pg space     131072     108647

               work       pers       clnt
pin             6951          0          0
in use        105067      25875          0


               size      inuse       free        pin    virtual
```

```
memory       131063     130951        113       6942     208406
pg space     131072     110432

               work       pers       clnt
pin            6951          0          0
in use       105127      25824          0
```

The output showing the top ten memory-using processes using the **svmon** command appears as follows:

```
# svmon -Pu -t 10

------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   12478 memory         92498     1259    95911   189707        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
  7a80        5 work shmat/mmap            52911     0  222 53058   0..65285
  d18a        3 work shmat/mmap            31670     0 33881 65535   0..65535
  4358        4 work shmat/mmap             4963     0 60572 65535   0..65535
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  8811        d work shared library text     274     0   24   393   0..65535
   c93        8 work shmat/mmap              244     0   12   256   0..65288
  e6ec        7 work shmat/mmap              240     0   16   256   0..65287
  5d79        6 work shmat/mmap              234     0   22   256   0..65286
  e28c        9 work shmat/mmap              232     0   24   256   0..65289
  735e        a work shmat/mmap              200     0   55   255   0..65034
  767c        2 work process private           4     1    3     5
65314..65535
  3e76        f work shared library data       3     0    4     5   0..709
  634c        1 pers code,/dev/hd3:21          1     0    -     -   0..2

------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
13796 dd              2829     1260     1100     4303        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  dc53        - pers /dev/hd3:45            1011     0    -     -   0..1010
  8811        d work shared library text     274     0   24   393   0..65535
  edae        2 work process private           8     1    0     8   0..20 :

65310..65535
  83b0        1 pers code,/dev/hd2:4164        6     0    -     -   0..5
  dbb3        f work shared library data       5     0    0     2   0..797
```

```
  949a        3 work shmat/mmap               1     1    0     1   0..0
  ac7d        4 work shmat/mmap               1     0    0     1   0..0
  ac5d        5 work shmat/mmap               1     0    0     1   0..0

-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   14856 dd              2826     1260     1100     4301        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
65475..65535
  dc53        - pers /dev/hd3:45            1011     0    -     -   0..1010
  8811        d work shared library text     274     0   24   393   0..65535
  83b0        1 pers code,/dev/hd2:4164        6     0    -     -   0..5
  6ce5        2 work process private           6     1    0     6   0..19 :

65310..65535
  5cc3        f work shared library data       4     0    0     2   0..797
  949a        3 work shmat/mmap                1     1    0     1   0..0
  ac7d        4 work shmat/mmap                1     0    0     1   0..0
  ac5d        5 work shmat/mmap                1     0    0     1   0..0

-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
    4994 ksh             1975     1259     1100     4400        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  8811        d work shared library text     274     0   24   393   0..65535
  7b7c        2 work process private          98     1    0    96   0..115 :

65310..65535
  e03c        1 pers code,/dev/hd2:4204       55     0    -     -   0..58
  c72a        f work shared library data      24     0    0    14   0..797
2865        - pers /dev/hd2:32343             2     0    -     -   0..1
  4b89        - pers /dev/hd2:10340             0     0    -     -   0..10

-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   15434 ksh             1897     1259     1100     4328        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
```

```
  8811        d work shared library text     274     0   24   393   0..65535
  e03c        1 pers code,/dev/hd2:4204       55     0    -     -   0..58
  92c3        2 work process private          29     1    0    28   0..94 :

65310..65535
  30f7        f work shared library data      15     0    0    10   0..797
  2865        - pers /dev/hd2:32343            2     0    -     -   0..1
  536a        - pers /dev/hd2:4522             0     0    -     -   0..7
   c91        - pers /dev/hd3:29               0     0    -     -

------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   16728 -ksh            1897     1259     1103     4324        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  8811        d work shared library text     274     0   24   393   0..65535
  e03c        1 pers code,/dev/hd2:4204       55     0    -     -   0..58
  ef7a        2 work process private          24     1    2    23   0..83 :

65310..65535
  8717        f work shared library data      18     0    1    11   0..382
  2865        - pers /dev/hd2:32343            2     0    -     -   0..1
  a2f4        - pers /dev/hd4:792              1     0    -     -   0..1
  f96d        - pers /dev/hd3:40               1     0    -     -   0..0

------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   15808 ksh            1896     1259     1166     4366        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  8811        d work shared library text     274     0   24   393   0..65535
  e03c        1 pers code,/dev/hd2:4204       55     0    -     -   0..58
cec9        2 work process private          25     1   54    62   0..91 :

65310..65535
  1752        f work shared library data      17     0   12    14   0..797
  2865        - pers /dev/hd2:32343            2     0    -     -   0..1
  e35c        - pers /dev/hd1:19               1     0    -     -   0..0

------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
    2622 X              1888     1268     1889     5132        N        N
```

```
  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  8811        d work shared library text     274     0   24   393   0..65535
  8971        2 work process private          52     8  712   763   0..825 :

65309..65535
  9172        1 pers code,/dev/hd2:18475      28     0    -     -   0..706
  fa9f        - work                           9     0   32    32   0..32783
  3987        3 work shmat/mmap                2     2    2     4   0..32767
  b176        f work shared library data       1     0   39    39   0..310
  e97d        - pers /dev/hd2:20486            0     0    -     -   0..7
  d97b        - pers /dev/hd3:25               0     0    -     -
3186        - work                           0     0    2     2   0..32768
   180        - pers /dev/hd2:20485            0     0    -     -   0..0
  4168        - pers /dev/hd9var:2079          0     0    -     -   0..0
  1963        - pers /dev/hd9var:2078          0     0    -     -   0..0
  90b2        - work                           0     0    2     2   0..32768
  9092        - pers /dev/hd4:2                0     0    -     -   0..0

-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   11402 -ksh            1882     1259     1166     4364        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  8811        d work shared library text     274     0   24   393   0..65535
  e03c        1 pers code,/dev/hd2:4204       55     0    -     -   0..58
  6b0d        2 work process private          18     1   52    59   0..83 :

65310..65535
  4328        f work shared library data      11     0   14    15   0..382
  2865        - pers /dev/hd2:32343            2     0    -     -   0..1
  3326        - pers /dev/hd4:605              0     0    -     -   0..1

-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
    5686 -ksh            1872     1259     1106     4304        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             1522  1258 1076  3897   0..32767 :

65475..65535
  8811        d work shared library text     274     0   24   393   0..65535
  e03c        1 pers code,/dev/hd2:4204       55     0    -     -   0..58
  72ee        2 work process private          12     1    5    12   0..82 :
```

```
65310..65535
  6aed        f work shared library data       6     0    1     2   0..382
  2865        - pers /dev/hd2:32343            2     0    -     -   0..1
  a2f4        - pers /dev/hd4:792              1     0    -     -   0..1
```

A snapshot of the paging space at various intervals using the command:

```
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
hd6         hdisk0            rootvg         512MB      95     yes   yes    lv
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
hd6         hdisk0            rootvg         512MB      97     yes   yes    lv
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
hd6         hdisk0            rootvg         512MB       9     yes   yes    lv
```

The output of the **vmtune** command:

```
# vmtune
vmtune:  current values:
  -p       -P        -r          -R         -f       -F       -N        -W
minperm  maxperm  minpgahead maxpgahead  minfree  maxfree  pd_npages maxrandwrt
  26007   104028       2          8        120      128     524288        0

  -M      -w      -k      -c        -b         -B           -u        -l    -d
maxpin npswarn npskill numclust numfsbufs hd_pbuf_cnt lvm_bufcnt lrubucket
defps
104851    4096    1024       1      93         80          9      131072     1

        -s              -n         -S           -h
sync_release_ilock  nokillroot  v_pinshm  strict_maxperm
        0               0           0             0

number of valid memory pages = 131063   maxperm=79.4% of real memory
maximum pinable=80.0% of real memory    minperm=19.8% of real memory
number of file memory pages = 13516     numperm=10.3% of real memory
```

Display the number of processors using the **lsdev** command:

```
# lsdev -Ccprocessor
proc0 Available 00-00 Processor
```

This is a single processor system.

7.4.2 Data analysis

From the output in the previous section, an investigation can be done on the various components within the system to determine the areas that are causing

performance problems. For this investigation, the output will mostly be from the **vmstat** command output. The other output is for information and confirmation.

The kthr (kernel thread) column

The kthr column provides information about the average number of threads on various queues.

Both the r and b counts are low, indicating that the system is executing the runable threads in the kernel sufficiently. The contention for CPU resources is low.

The memory column

The memory column displays information about the use of real and virtual memory. A page is 4096 bytes in size.

In the avm column it can be seen that the average number of pages allocated increased. The system will keep allocating pages until all paging space available is used (check with the **lsps -a** command). When all the paging space has been utilized or reaches 100 percent, the system will start killing processes to make paging space available for use.

The fre column shows the average number of free memory pages. The MINFREE value for this system is 120, as shown with the **vmtune** command. In the example, the free memory stayed around the MINFREE value until it dropped to below 100 and almost to 0. This is one of the indications that the system was thrashing.

The page column

The page column displays information about page faults and paging activity. These averages are given per second. Paging space is the part of virtual memory that resides on disk. It is used as an overflow when memory is overcommitted. Paging consists of paging logical volumes dedicated to the storage of working set pages that have been stolen from real memory. When a stolen page is referenced by the process, a page fault occurs and the page must be read into memory from paging space. Whenever a page of working storage is stolen, it is written to paging space. If not referenced again, it remains on the paging device until the process terminates or disclaims the space. Subsequent references to addresses contained within the faulted-out pages result in page faults, and the pages are paged in individually by the system. When a process terminates normally, any paging space allocated to that process is freed.

The re column, which is the number of reclaimed pages, remained at 0 throughout.

The pi column varied from 0 to the highest level of 56 pages paged in from paging space. Although a pi level of no more than 5 is considered acceptable, a level higher than 5 is not necessarily an indication of a performance problem, due to the fact that for every page paged in, there must have been a page that was paged out.

The po column, which reflects the number of pages paged out, was between 183 and 445 per second. With a high rate of paging, the system may see some performance degradation, as the paging space is kept on the hard disk and is accessed slower than RAM.

The fr column is the number of pages freed in a second.

The sr column is the number of pages scanned by the page placement algorithm. If the ratio between fr:sr is high, this can indicate a memory constraint, for example, if the ratio is 1:3, it will mean that for every page freed, three will need to be checked. In the example, the ratio is close to 1:2.

The faults column

The information under the faults heading displays the trap and interrupt rate averages per second.

The in column is the number of device interrupts per second and is always greater than 100.

The sy column is the number of system calls and it is extremely difficult to say what this figure should be.

The cs column is the number of context switches.

The cpu column

The information under the cpu heading provides a breakdown of CPU usage.

The us column indicates the amount of time spent in user mode. In the example, this is never above 5 percent.

The sy column indicates the amount of time spent in system mode. In this example, it is never below 92 percent.

The id column indicates the amount of idle time. In this example, the cpu is never idle.

The wa column indicates the amount of idle time with pending local disk I/O. In the example, it is never higher than seven percent.

Note: In a single processor system, if us + sy is greater than 90 percent, the system can be considered CPU bound.

7.4.3 Recommendation

From the data, the following are some recommendations that can be implemented to ease the situation:

- If it is possible, an additional CPU might be added to try and get the CPU utilization below 80 percent.
- Add an additional paging space on another internal disk. The reason for adding this paging area to an internal disk is for speed and availability. It is advisable to always spread the paging space across multiple disks, as this will improve paging performance.
- If this is a situation that occurs only at certain times, it may be possible to reschedule large jobs and spread them out, so as not to conflict with one another.

8

Scenario assessment quiz

In this chapter you will be provided with two scenarios to examine and questions to answer based on those scenarios.

8.1 Scenario one

Robutz, Inc. is a small startup robot toy manufacturing company. They decided to purchase an RS/6000 F50 to keep track of their suppliers, inventory, production schedules as well as marketing support. The system they purchased consists of the following components:

- F50 running AIX Version 4.3.2
- Single 166 MHz PowerPC 604e
- RAM 128 MB
- 3–4.5 GB SCSI Disk Drives
- 1–8 mm Tape Drive
- Integrated 10 Mbps Ethernet will be used to connect to their network.
- They anticipate seven users to be using the system, connecting with their PCs over the network.
- They will be using an off-the-shelf program with Sybase as their database.
- They will have two network printers.
- They have decided to create three Volume Groups (each on separate disk): rootvg, sybasevg, and dumpvg.

Scenario questions:

1. Using a customer's **vmtune** settings, which of the following parameters should be changed to help eliminate paging?

```
# vmtune
vmtune:  current values:
  -p       -P        -r          -R         -f       -F       -N        -W
minperm  maxperm  minpgahead maxpgahead  minfree  maxfree  pd_npages
maxrandwrt
   6344    25376       2          8        120      128     524288         0

  -M      -w      -k      -c        -b         -B          -u        -l
maxpin npswarn npskill numclust numfsbufs hd_pbuf_cnt lvm_bufcnt lrubucket
defps
 26196    1024     256       1      93         64          9      131072
0

        -s            -n           -S           -h
sync_release_ilock  nokilluid    v_pinshm  strict_maxperm
        0             0            0              0

number of valid memory pages = 32744    maxperm=77.5% of real memory
maximum pinable=80.0% of real memory    minperm=19.4% of real memory
```

```
number of file memory pages = 24067     numperm=73.5% of real memory
```

A. maxpin

B. maxfree

C. maxperm

D. maxpgahead

2. Using the vmtune output, as shown in the exhibit, which of the following is the most probable cause for the high paging space allocation?

```
# vmtune
vmtune:  current values:
  -p       -P        -r          -R         -f       -F       -N        -W
minperm  maxperm  minpgahead maxpgahead  minfree  maxfree  pd_npages
maxrandwrt
   6344    25376       2          8        120      128     524288        0

  -M      -w      -k      -c        -b         -B          -u        -l
maxpin npswarn npskill numclust numfsbufs hd_pbuf_cnt lvm_bufcnt lrubucket
defps
 26196    1024     256       1      93         64          9      131072
0

        -s            -n          -S           -h
sync_release_ilock  nokilluid    v_pinshm  strict_maxperm
        0             0            0              0

number of valid memory pages = 32744    maxperm=77.5% of real memory
maximum pinable=80.0% of real memory    minperm=19.4% of real memory
number of file memory pages = 24067     numperm=73.5% of real memory
```

A. defps is 0.

B. minpgahead is 2.

C. maxpgahead is 8.

D. maxfree is 128.

8.1.1 Answers

The following are the preferred answers to the questions provided in this section.

1. C
2. A

8.2 Scenario two

MooCow Inc. has a single processor F50 with 512 MB of memory and four 4.5 GB SCSI drives. They have a flat IP network using the built-in 10 Mbps Ethernet adapter and are running AIX Version 4.3.3 with all the latest PTFs. The system administrator reboots the system every Sunday after their full-weekly backup. Throughout the week, the users notice degrading performance with the application A response time.

1. Which of the following tools should be used to begin gathering historical performance data on the system for a performance analysis?

   A. **sar**

   B. **iostat**

   C. **netstat**

   D. **vmstat**

2. MooCow Inc. has added a CPU, increased physical memory, and installed another two SCSI drives in order to support application B. Since application B has been installed, users are complaining about application response times during peak business hours.

   Use the information provided in Figure 8-1, Figure 8-2 on page 279, Figure 8-3 on page 279, and Figure 8-4 on page 280 to help you with your answer.

```
#  ps -ef | sort +3 -r | head
     UID    PID   PPID   C    STIME    TTY   TIME CMD
   user1  84964  85996 112 14:25:43      -   4:04 /usr/app/exp_report
   user1 101194      1  38 14:10:16 pts/60   2:01 /opt/lotus/notes/la
test/ibmpow/notes
    root  91686  90358  37 16:53:28      -   0:02 /opt/lotus/notes/la
test/ibmpow/notes
    root  98544  99676  28 14:34:38 pts/59   0:00 ps -ef
   user2  61182  62462  18 18:41:19      - 105:12 /usr/netscape/commu
nicator/bidi/netscape_aix4
   user3  83414      1   7 09:28:52 pts/44   4:18 /opt/lotus/notes/la
test/ibmpow/notes
    root  73560  71826   6 06:56:43 pts/31  12:24 java com/ibm/retain
/client/ncr/NCRetain -pf austin_ibm_com
   user4 100330      1   5 14:07:50 pts/41   0:23 /opt/lotus/notes/la
test/ibmpow/notes
   user3  56514      1   4 17:15:41      -  42:03 SoftWindows95
    root  91182      1   4 10:45:54 pts/21   3:10 /opt/lotus/notes/la
test/ibmpow/notes
```

Figure 8-1 The ps command output

```
# vmstat l20 l0
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b    avm   fre  re  pi  po fr  sr cy  in    sy  cs us sy id wa
 5  1  19331   824   0   0   0  0   0  0 636 1955 674 81 19  0  0
 4  1  19803   996   0   0   0  0   0  0 533 7466 591 76 24  0  0
 5  1  19974   860   0   0   0  0   0  0 822 4055 892 81 19  0  0
 5  1  19815   860   0   0   0  0   0  0 535 4096 509 71 29  0  0
 2  1  19816   855   0   0   0  0   0  0 577 4582 598 83 17  0  0
 3  1  19816   737   0   0   0  0   0  0 602 2720 672 66 34  0  0
 2  1  19895   724   0   0   0  0   0  0 616 3842 698 82 18  0  0
 5  1  17147   724   0   0   0  0   0  0 649 6427 626 75 25  0  0
 3  1  17065   720   0   0   0  0   0  0 516 3629 543 78 22  0  0
 5  1  17163   720   0   0   0  0   0  0 614 9030 688 64 35  0  0
 3  1  17343   720   0   0   0  0   0  0 420 8777 487 71 29  0  0
 5  1  17579   712   0   0   0  0   0  0 466 2182 473 62 38  0  0
 2  1  17647   712   0   0   0  0   0  0 497 3298 310 62 37  0  0
```

Figure 8-2 The vmstat command output

```
# iostat l20 5
tty:   tin   tout   avg-cpu:  % user    % sys    % idle    % iowait
       0.0  523.5               81.2     19.3       0.1         0.4

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.3       7.6       2.6          0       106
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
hdisk4           0.0       0.0       0.0          0         0
hdisk5           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:   tin   tout   avg-cpu:  % user    % sys    % idle    % iowait
       0.0  529.2               76.4     23.6       0.0         0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
hdisk4           0.0       0.0       0.0          0         0
hdisk5           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:   tin   tout   avg-cpu:  % user    % sys    % idle    % iowait
       0.0  523.4               80.6     19.4       0.0         0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
hdisk4           0.0       0.0       0.0          0         0
hdisk5           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0
```

Figure 8-3 The iostat command output

```
# netstat -m

49 mbufs in use:
28 mbuf cluster pages in use
124 Kbytes allocated to mbufs
0 requests for mbufs denied
0 calls to protocol drain routines

Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed    free   hiwat   freed
32              111      2757      0      17     640       0
64               93     79910      0      35     320       0
128              65    605956      0     127     160     313
256              89  84401838      0     151     384       0
512              58   1122650      0      14      40      52
1024              0    385298      0      20      20       0
2048              1    771690      0       9      10       1
4096             24   2356796      0      31     120       0
8192              1     99962      0       9      10       0
16384             0     32157      0      20      24       7

******* CPU 1 *******
By size       inuse     calls failed    free   hiwat   freed
32               19      2602      0     109     640       0
64               63     55968      0      65     320       0
128              54    578108      0     138     160     328
256              45  79776714      0     179     384       0
512              27    954403      0      29      40      28
1024              0    369263      0      20      20       0
2048              0    752292      0      10      10       2
4096              4   2203880      0      28     120       0
8192              2     95150      0       6      10       0
16384             1     29738      0      20      24       8

By type       inuse     calls failed  memuse  memmax  mapb
mbuf             47 285532098      0   12032   59648     0
mcluster         27  17060009      0  110592  301056     0
socket          203    364802      0   33728   50176     0
pcb             167    318578      0   27200   46400     0
routetbl          8        20      0    1312    2080     0
fragtbl           0      8848      0       0      96     0
ifaddr            7         9      0     832     832     0
mblk             24   2550913      0    3712   90368     0
mblkdata          7     48967      0   50176  126976     0
strhead          46      1333      0    9152   15744     0
strqueue         45      3261      0   23040   39424     0
strmodsw         21        21      0    1344    1344     0
strpoll           0         1      0       0      32     0
strosr            0      3957      0       0     256     0
strsyncq         53     10352      0    6208   10368     0
streams         201      4044      0   24416   33376     0
kernel table     43    143395      0   60768   86368     0
temp              7        52      0   14528   15040     0

Streams mblk statistic failures:
0 high priority mblk failures
0 medium priority mblk failures
0 low priority mblk failures
```

Figure 8-4 The netstat command output

Using information provided in the exhibits, which of the following conclusions is most appropriate to draw about the problem on the system?

A. The system is CPU bound.

B. The system is I/O bound.

C. The system is memory bound.

D. The system is network bound.

3. Based on the statistics, which of the following reports helped to determine the problem?

   A. The stime from the **ps** report

   B. The % user and % sys metrics from the **iostat** report

   C. The statistics from the b column in the **vmstat** report

   D. The number of requests for mbufs denied from the **netstat** report

4. Which of the following procedures should be performed to remedy the situation?

   A. Stripe the logical volumes.

   B. Increase the tcp_sendspace to 65536.

   C. Use **vmtune** to change the file system cache.

   D. Move the /usr/app/exp_report process to off-peak hours.

5. The actions taken so far have only increased performance slightly. The service level agreement is still not being achieved. Which of the following procedures should be performed?

   A. Install a PCI SCSI-RAID adapter.

   B. Install another 512 MB of memory.

   C. Upgrade to a four-way processor.

   D. Upgrade to a 10/100 Ethernet adapter.

6. Which of the following should be implemented to balance available CPU/memory resources between applications?

   A. Loadleveler

   B. Nice/renice

   C. Resource Manager

   D. Workload Manager

7. MooCow Inc. is still experiencing performance problems. Using the output shown in Figure 8-5, which of the following conclusions is most appropriate to draw?

```
# vmstat l20 l0
kthr    memory             page            faults        cpu
----- ---------- -------------------- ------------ -----------
r  b   avm   fre re  pi  po  fr sr cy  in   sy  cs us sy id wa
0  1 19331  824  0   0   0   0   0  0 636 1955 674  0  0 99  0
0  1 19803  996  0   0   0   0   0  0 533 7466 591  0  0 99  0
0  1 19974  860  0   0   0   0   0  0 822 4055 892  0  0 99  0
0  1 19815  860  0   0   0   0   0  0 535 4096 509  0  0 99  0
0  1 19816  855  0   0   0   0   0  0 577 4582 598  0  0 99  0
0  1 19816  737  0   0   0   0   0  0 602 2720 672  0  0 99  0
0  1 19895  724  0   0   0   0   0  0 616 3842 698  0  0 99  0
0  1 17147  724  0   0   0   0   0  0 649 6427 626  0  0 99  0
0  1 17065  720  0   0   0   0   0  0 516 3629 543  0  0 99  0
0  1 17163  720  0   0   0   0   0  0 614 9030 688  0  0 99  0
0  1 17343  720  0   0   0   0   0  0 420 8777 487  0  0 99  0
0  1 17579  712  0   0   0   0   0  0 466 2182 473  0  0 99  0
0  1 17647  712  0   0   0   0   0  0 497 3298 310  0  0 99  0

# iostat l20 5
tty:  tin     tout   avg-cpu:  % user    % sys    % idle    % iowait
      0.0      0.0                0.0      0.0      99.0        0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
hdisk4           0.0       0.0       0.0          0         0
hdisk5           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:  tin     tout   avg-cpu:  % user    % sys    % idle    % iowait
      0.0      0.0                0.0      0.0      99.0        0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
hdisk4           0.0       0.0       0.0          0         0
hdisk5           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:  tin     tout   avg-cpu:  % user    % sys    % idle    % iowait
      0.0      0.0                0.0      0.0      99.0        0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
hdisk4           0.0       0.0       0.0          0         0
hdisk5           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0
```

Figure 8-5 The vmstat and iostat command outputs

A. The system is CPU bound.

B. The system is I/O bound.

C. The system is memory bound.

D. The system is network bound.

8. Given the details of the exhibit, which of the following procedures should be performed to fix the bottleneck?

   A. Add CPU(s).

   B. Add 512 MB of memory.

   C. Stripe the logical volumes.

   D. Implement a switched Ethernet.

9. The backups for a system are being taken over the network to a TSM Server. Unfortunately, they are taking too long and the service level agreement is not being met with the customer. Based upon the current configuration and details shown in the exhibit, which of the following tuning procedures should be performed to improve the file system's backup performance?

   A. Increase the maxfree value using **vmtune**.

   B. Increase the mult value using **schedtune**.

   C. Increase the queue length on the disk drives using **chdev**.

   D. Increase the tcp_sendspace and tcp_recvspace on your system and the TSM Server.

10. The backups are still not meeting the service level agreement. It has been suggested that you to re-examine the I/O performance.

```
# lsps -s

Total Paging Space   Percent Used
      100MB              37%

# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto
Type
hd6         hdisk0            rootvg         100MB      38     yes   yes
lv

# lsvg
rootvg
datavg


# lspv
hdisk0        000038744c632197    rootvg
```

```
hdisk1          00002199abf65a1a    datavg
hdisk2          00000228b9c5d7da    datavg
hdisk3          00002199b40b728c    datavg

# lspv -l hdisk0
hdisk0:
LV NAME               LPs   PPs   DISTRIBUTION          MOUNT POINT
hd5                   2     2     02..00..00..00..00    N/A
hd3                   6     6     02..00..04..00..00    /tmp
hd2                   117   117   00..47..42..28..00    /usr
hd8                   1     1     00..00..01..00..00    N/A
hd4                   2     2     00..00..02..00..00    /
hd9var                1     1     00..00..01..00..00    /var
hd6                   128   128   29..50..49..00..00    N/A
hd1                   3     3     00..00..01..02..00    /home
lv00                  10    10    00..00..00..10..00    /database


# lspv -l hdisk1
hdisk1:
LV NAME               LPs   PPs   DISTRIBUTION          MOUNT POINT
loglv00               1     1     01..00..00..00..00    N/A
lv01                  10    10    00..00..00..00..10    /db01
lv02                  10    10    00..00..00..00..10    /db02
lv03                  10    10    10..00..00..00..00    /db03


# lspv -l hdisk2
hdisk2:
LV NAME               LPs   PPs   DISTRIBUTION          MOUNT POINT


# lspv -l hdisk3
hdisk3:
LV NAME               LPs   PPs   DISTRIBUTION          MOUNT POINT

# lslv lv01
LOGICAL VOLUME:     lv01                   VOLUME GROUP:   datavg
LV IDENTIFIER:      0000881962b29b51.1     PERMISSION:     read/write
VG STATE:           active/complete        LV STATE:       opened/syncd
TYPE:               jfs                    WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        8 megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                1                      PPs:            1
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    yes
INTRA-POLICY:       middle                 UPPER BOUND:    32
```

```
LOGICAL VOLUME:     lv01                    VOLUME GROUP:     datavg
MOUNT POINT:        /db01                   LABEL:            /db01
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes

# lslv lv02
LOGICAL VOLUME:     lv02                    VOLUME GROUP:     datavg
LV IDENTIFIER:      0000881962b29b51.3      PERMISSION:       read/write
VG STATE:           active/complete         LV STATE:         opened/syncd
TYPE:               jfs                     WRITE VERIFY:     off
MAX LPs:            512                     PP SIZE:          8 megabyte(s)
COPIES:             1                       SCHED POLICY:     parallel
LPs:                1                       PPs:              1
STALE PPs:          0                       BB POLICY:        relocatable
INTER-POLICY:       minimum                 RELOCATABLE:      yes
INTRA-POLICY:       middle                  UPPER BOUND:      32
MOUNT POINT:        /db02                   LABEL:            /db02
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes

# lslv lv03
LOGICAL VOLUME:     lv03                    VOLUME GROUP:     datavg
LV IDENTIFIER:      0000881962b29b51.4      PERMISSION:       read/write
VG STATE:           active/complete         LV STATE:         opened/syncd
TYPE:               jfs                     WRITE VERIFY:     off
MAX LPs:            512                     PP SIZE:          8 megabyte(s)
COPIES:             1                       SCHED POLICY:     parallel
LPs:                1                       PPs:              1
STALE PPs:          0                       BB POLICY:        relocatable
INTER-POLICY:       minimum                 RELOCATABLE:      yes
INTRA-POLICY:       middle                  UPPER BOUND:      32
MOUNT POINT:        /db03                   LABEL:            /db03
MIRROR WRITE CONSISTENCY: on
EACH LP COPY ON A SEPARATE PV ?: yes
```

Using information provided in the exhibits, which of the following procedures should be performed to improve the I/O performance?

A. Change the stripe size to 32 KB.

B. Adjust the value of maxrndwrt.

C. Upgrade your disk subsystem to RAID 5.

D. Move each file system to separate disks.

11. The actions taken so far have helped, but more actions are necessary. Which of the following procedures should be performed to accelerate performance of the I/O?

A. Disable the write consistency check.

B. Add a JFS log for each file system.

C. Increase the maxpin parameter with vmtune.

D. Increase the syncd frequency from 60 to 10.

8.2.1 Answers

The following are the preferred answers to the questions provided in this section.

1. A
2. A
3. B
4. D
5. C
6. D
7. D
8. D
9. D
10. D
11. B

A

The error log

The following topics are discussed in this appendix:

- A general discussion about the error logging subsystem
- How to manage error logs
- How to read error logs

The error log is the first place where an administrator will search for the cause of improper system performance.

Overview

The error-logging process begins when an operating system module detects an error. The error-detecting segment of code then sends error information to either the errsave and errlast kernel services or the errlog application subroutine where the information is, in turn, written to the /dev/error special file. This process then adds a timestamp to the collected data. The errdemon daemon constantly checks the /dev/error file for new entries, and when new data is written, the daemon conducts a series of operations.

Before an entry is written to the error log, the errdemon daemon compares the label sent by the kernel or application code to the contents of the *error record template repository*. If the label matches an item in the repository, the daemon collects additional data from other parts of the system.

The system administrator can look at the error log to determine what caused a failure, or to periodically check the health of the system when it is running.

The software components that allow the AIX kernel and commands to log errors to the error log are contained in the fileset bos.rte.serv_aid. This fileset is automatically installed as part of the AIX installation process.

The commands that allow you to view and manipulate the error log, such as the **errpt** and **errclear** commands, are contained in the fileset called bos.sysmgt.serv_aid. This fileset is not automatically installed by earlier releases of AIX Version 4. Use the following command to check whether the package is installed on your system:

```
# lslpp -l bos.sysmgt.serv_aid
  Fileset                      Level  State      Description
----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.sysmgt.serv_aid        4.3.3.0  COMMITTED  Software Error Logging and
                                                 Dump Service Aids

Path: /etc/objrepos
  bos.sysmgt.serv_aid        4.3.3.0  COMMITTED  Software Error Logging and
                                                 Dump Service Aids
```

Managing the error log

Error logging is automatically started during system initialization by the /sbin/rc.boot script and is automatically stopped during system shutdown by the shutdown script. The part of /sbin/rc.boot that starts error logging looks like:

```
if [ -x /usr/lib/errdemon ]
```

```
then
        echo "Starting the error daemon" | alog -t boot
        /usr/bin/rm -f /tmp/errdemon.$$
        /usr/lib/errdemon >/tmp/errdemon.$$ 2>&1
        if [ $? -ne 0 ]
        then
                cat /tmp/errdemon.$$ | alog -t boot
                echo "Starting the errdemon with the system default" \
                        "log file, /var/adm/ras/errlog." | alog -t boot
                /usr/lib/errdemon -i /var/adm/ras/errlog
        fi
        /usr/bin/rm -f /tmp/errdemon.$$
fi
```

As you can see, the **/usr/lib/errdemon** command starts error logging and initializes /var/adm/ras/errlog as a default log file.

Configuring error log

You can customize the name and location of the error log file and the size of the internal error buffer to suit your needs.

To list the current settings, run the **/usr/lib/errdemon -l** command. The values for the error log file name, error log file size, and buffer size that are currently stored in the error log configuration database are displayed on your screen.

```
# /usr/lib/errdemon -l
Error Log Attributes
-------------------------------------------
Log File                /var/adm/ras/errlog
Log Size                1048576 bytes
Memory Buffer Size      8192 bytes
```

You can change all of values listed above:

- To change the name of the file used for error logging, run:

  ```
  # /usr/lib/errdemon -i /var/adm/ras/errlog.new
  ```

  The /var/adm/ras/errlog.new file name is saved in the error log configuration database and the error daemon is immediately restarted.

- To change the maximum size of the error log file to 2000000 bytes, type:

  ```
  # /usr/lib/errdemon -s 2000000
  ```

  The specified log file size limit is saved in the error log configuration database and the error daemon is immediately restarted.

- To change the size of the error log device driver's internal buffer to 16384 bytes, enter:

```
# /usr/lib/errdemon -B 16384
0315-175 The error log memory buffer size you supplied will be rounded up
to a multiple of 4096 bytes.
```

  The specified buffer size is saved in the error log configuration database and, if it is larger than the buffer size currently in use, the in-memory buffer is immediately increased. The size you specify is rounded up to the next integral multiple of the memory page size (4 KBs).

  Note: The memory used for the error log device driver's in-memory buffer is not available for use by other processes (the buffer is pinned).

Now verify what you did with the following command. Note the highlighted values reflect the changes you made:

```
# /usr/lib/errdemon -l
Error Log Attributes
------------------------------------------
Log File                /var/adm/ras/errlog.new
Log Size                2000000 bytes
Memory Buffer Size      16384 bytes
```

Clearing the error log

Clearing of the error log implies deleting old or unnecessary entries from the error log. Clearing is normally done as part of the daily **cron** command execution. To check it, type:

```
# crontab -l | grep errclear
0 11 * * * /usr/bin/errclear -d S,O 30
0 12 * * * /usr/bin/errclear -d H 90
```

If it is not done automatically, you should probably clean the error log regularly.

To delete all the entries from the error log, use the following command:

```
# errclear 0
```

To selectively remove entries from the error log, for example, to delete all software error entries, use the following command:

```
# errclear -d S 0
```

To selectively remove entries from the error log, for example, to delete all hardware error entries, use the following command:

```
# errclear -d H 0
```

Alternatively, use the **smitty errclear** command.

Reading error logs in details

You can generate an error report from data collected in the error log. There are two main ways to view the error log:

- The easiest way to read the error log entries is with the **smitty errpt** command. Output from this command is shown in Figure A-1.

```
                              Generate an Error Report

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
  CONCURRENT error reporting?                          yes
  SUMMARY or DETAILED error report                     summary                +
  Error CLASSES (default is all)                       [H]                    +
  Error TYPES   (default is all)                       [TEMP]                 +
  Error LABELS (default is all)                        []                     +
  Error ID's      (default is all)                     []                     +X
  Resource CLASSES (default is all)                    []
  Resource TYPES   (default is all)                    []
  Resource NAMES  (default is all)                     []
  SEQUENCE numbers (default is all)                    []
  STARTING time interval                               []
  ENDING time interval                                 []
  LOGFILE                                              [/var/adm/ras/errlog]
[MORE...3]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure A-1 smitty errpt output

- The second way to display error log entries is with the **errpt** command. It allows flags for selecting errors that match specific criteria. By using the default condition, you can display error log entries in the reverse order they occurred and were recorded.

The errpt command output

By using the -c flag, you can display errors as they occur. The default summary report contains one line of data for each error:

```
# errpt | pg
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
2BFA76F6   0627172400 T S SYSPROC        SYSTEM SHUTDOWN BY USER
9DBCFDEE   0627172700 T O errdemon       ERROR LOGGING TURNED ON
```

```
192AC071   0627172300 T O errdemon       ERROR LOGGING TURNED OFF
1581762B   0627132600 T H cd0            DISK OPERATION ERROR
1581762B   0627132000 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
E18E984F   0627100000 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0627095400 P S SRC            SOFTWARE PROGRAM ERROR
```

The fields used in this report are discussed in the following sections.

Identifier

Numerical identifier for the event.

Timestamp

Time when the error occurs in format mmddhhmmyy. The timestamp 0627172400 indicates that the error occur June 27th at 17:24 (5:24 p.m.) year 00 (year 2000).

Type

Severity of the error that has occurred. There are six possible values:

PEND The loss of availability of a device or component is imminent.

PERF The performance of the device or component has degraded to below an acceptable level.

PERM A condition has occurred that could not be recovered from. Error types with this value are usually the most severe errors and are more likely to mean that you have a defective hardware device or software module. Error types other than PERM usually do not indicate a defect, but they are recorded so that they can be analyzed by the diagnostics programs.

TEMP A condition occurred that was recovered from after a number of unsuccessful attempts.

UNKN It is not possible to determine the severity of the error.

INFO The error log entry is informational and was not the result of an error.

Class

General source of the error. The possible error classes are:

H Hardware. When you receive a hardware error, refer to your system operator guide for information about performing diagnostics on the problem device or other piece of equipment.

S Software.

O Informational messages.

U Undetermined (for example, network).

Resource name

For software errors, this is the name of a software component or an executable program. For hardware errors, this is the name of a device or system component. It is used to determine the appropriate diagnostic modules that are to be used to analyze the error.

Description

A brief summary of the error.

Formatted output from errpt command

The following list provides a series of format options for the **errpt** command.

- To list all hardware errors, enter:

```
# errpt -d H
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
1581762B   0627132600 T H cd0            DISK OPERATION ERROR
1581762B   0627132000 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
5BF9FD4D   0615173700 T H tok0           PROBLEM RESOLVED
2A9F5252   0615161700 P H tok0           WIRE FAULT
2A9F5252   0615161600 P H tok0           WIRE FAULT
2A9F5252   0615161600 P H tok0           WIRE FAULT
5BF9FD4D   0615155900 T H tok0           PROBLEM RESOLVED
2A9F5252   0615151400 P H tok0           WIRE FAULT
2A9F5252   0615151300 P H tok0           WIRE FAULT
2A9F5252   0615151300 P H tok0           WIRE FAULT
2A9F5252   0615151300 P H tok0           WIRE FAULT
```

- To get a detailed report of all software errors, enter:

```
# errpt -a -d S | pg
---------------------------------------------------------------------------
LABEL:          REBOOT_ID
IDENTIFIER:     2BFA76F6

Date/Time:       Tue Jun 27 17:24:55
Sequence Number: 33
Machine Id:      000615142C00
Node Id:         server4
Class:           S
Type:            TEMP
Resource Name:   SYSPROC
```

```
Description
SYSTEM SHUTDOWN BY USER

Probable Causes
SYSTEM SHUTDOWN

Detail Data
USER ID
          0
0=SOFT IPL 1=HALT 2=TIME REBOOT
          0
TIME TO REBOOT (FOR TIMED REBOOT ONLY)
          0
---------------------------------------------------------------------
...
```

- To display a report of all errors logged for the error identifier E18E984F, enter:

```
# errpt -j E18E984F
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
E18E984F   0627100000 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0627095400 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0627093000 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0626182100 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0626181400 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0626130400 P S SRC            SOFTWARE PROGRAM ERROR
```

- To display a report of all errors that occur after June 26, 2000 at 18:14 (time), enter:

```
# errpt -s 0626181400
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
2BFA76F6   0627172400 T S SYSPROC        SYSTEM SHUTDOWN BY USER
9DBCFDEE   0627172700 T O errdemon       ERROR LOGGING TURNED ON
192AC071   0627172300 T O errdemon       ERROR LOGGING TURNED OFF
1581762B   0627132600 T H cd0            DISK OPERATION ERROR
1581762B   0627132000 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
E18E984F   0627100000 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0627095400 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0627093000 P S SRC            SOFTWARE PROGRAM ERROR
2BFA76F6   0627092700 T S SYSPROC        SYSTEM SHUTDOWN BY USER
9DBCFDEE   0627092900 T O errdemon       ERROR LOGGING TURNED ON
192AC071   0627092500 T O errdemon       ERROR LOGGING TURNED OFF
369D049B   0626183400 I O SYSPFS         UNABLE TO ALLOCATE SPACE IN FILE
SYSTEM
E18E984F   0626182100 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0626181400 P S SRC            SOFTWARE PROGRAM ERROR
```

- To obtain all the errors with resource name cd0 from the error log, enter:

```
# errpt -N cd0
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
1581762B   0627132600 T H cd0            DISK OPERATION ERROR
1581762B   0627132000 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
1581762B   0627131900 T H cd0            DISK OPERATION ERROR
```

- The -c flag formats and displays each of the error entries concurrently, that is, at the time they are logged. The existing entries in the log file are displayed in the order in which they were logged. An example below:

```
# errpt -c
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
9DBCFDEE   0823160702 T O errdemon       ERROR LOGGING TURNED ON
2BFA76F6   0823160002 T S SYSPROC        SYSTEM SHUTDOWN BY USER
FFE305EE   0823160702 P H tok0           WIRE FAULT
75CE5DC5   0823160702 I H tok0           ADAPTER ERROR
E18E984F   0823160802 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0823160802 P S SRC            SOFTWARE PROGRAM ERROR
E18E984F   0823160802 P S SRC            SOFTWARE PROGRAM ERROR
FFE305EE   0823160802 P H tok0           WIRE FAULT
```

Command summary

The following section discusses the **errpt** command.

The errpt command

The **errpt** command generates an error report from entries in an error log. The command has the following syntax:

```
errpt [ -a ] [ -c ] [ -d ErrorClassList ] [ -e EndDate ] [ -j ErrorID ]
[ -s StartDate ] [ -N ResourceNameList ] [ -S ResourceClassList ]
[ -T ErrorTypeList ]
```

The commonly used flags are listed in Table A-1.

Table A-1 Commonly used flags of the errpt command

Flag	Description
-a	Displays information about errors in the error log file in detailed format.
-c	Formats and displays each of the error entries concurrently, that is, at the time they are logged. The existing entries in the log file are displayed in the order in which they were logged.

Flag	Description
-d *ErrorClassList*	Limits the error report to certain types of error records specified by the valid ErrorClassList variables: H (hardware), S (software), 0 (errlogger command messages), and U (undetermined).
-e *EndDate*	Specifies all records posted prior to and including the EndDate variable.
-j *ErrorID*	Includes only the error-log entries specified by the ErrorID (error identifier) variable.
-s *StartDate*	Specifies all records posted on and after the StartDate variable.
-N *ResourceNameList*	Generates a report of resource names specified by the ResourceNameList variable. The ResourceNameList variable is a list of names of resources that have detected errors.
-S *ResourceClassList*	Generates a report of resource classes specified by the ResourceClassList variable.
-T *ErrorTypeList*	Limits the error report to error types specified by the valid ErrorTypeList variables: INFO, PEND, PERF, PERM, TEMP, and UNKN.

Quiz

The following assessment question helps verify your understanding of the topics discussed in this appendix.

1. Which of the following commands should be used to list the system errors logged following February 29, 2000?

   A. **`/usr/bin/errpt -s 0301000000`**

   B. **`/usr/bin/lssrc -s 0301000000`**

   C. **`/usr/lib/errdemon -s 0301000000`**

   D. **`/usr/sbin/syslogd -m 0301000000`**

Answers

The following is the preferred answer to the question provided in this section.

1. A

Exercises

The following exercises provide sample topics for self study. They will help ensure comprehension of this appendix.

1. Change the default error log attributes.
2. Using the **errpt** command, generate a report of errors caused by the errdemon resource.
3. Using the **errpt** command, generate a report of software errors, but limit it to temporary errors.
4. Generate the same reports using the **smitty** tool.
5. Delete all software logs.

B

Installing the performance tools

Anyone faced with the task of keeping a computer system well tuned and capable of performing as expected recognizes the following areas as essential for success:

Load monitoring Resource load must be monitored so performance problems can be detected as they occur or (preferably) predicted before they do.

Analysis and control Once a performance problem is encountered, the proper tools must be selected and applied so that the nature of the problem can be understood and corrective action taken.

Capacity planning Long-term capacity requirements must be analyzed so that sufficient resources can be acquired before they are required.

Tools and filesets

The intention of this section is to give you a list of all the performance tools discussed in this book together with the path that is used to call the command and the fileset the tool is part of.

Many of the performance tools are located in filesets that obviously would contain them, such as bos.perf.tools or perfagent.tools. However, some of them are located in filesets that are not quite as obvious. Common examples are **vmtune** or **schedtune**, which are both part of the bos.adt.samples fileset. You will often find that this fileset is not installed on a system because it does not contain performance tools.

Table B-1 lists the tools discussed in this book, their full path names, and their fileset information.

Table B-1 Commands or tools, path names, and filesets

Command or tool	Full path name	Fileset name/URL
3dmon	/usr/bin/3dmon	perfmgr.network
alstat	/usr/bin/alstat	bos.perf.tools
atmstat	/usr/bin/atmstat	devices.common.IBM.atm.rte
bindintcpu	/usr/sbin/bindintcpu	devices.chrp.base.rte
bindprocessor	/usr/sbin/bindprocessor	bos.mp
curt	-	ftp://software.ibm.com/
emstat	/usr/bin/emstat	bos.perf.tools
entstat	/usr/bin/entstat	devices.common.IBM.ethernet.rte
estat	/usr/lpp/ssp/css/css	ssp.css
fddistat	/usr/bin/fddistat	devices.common.IBM.fddi.rte
fdpr	/usr/bin/fdpr	perfagent.tools
filemon	/usr/bin/filemon	bos.perf.tools
fileplace	/usr/bin/fileplace	bos.perf.tools
genkex	/usr/bin/genkex	bos.perf.tools
genkld	/usr/bin/genkld	bos.perf.tools
genld	/usr/bin/genld	bos.perf.tools

Command or tool	Full path name	Fileset name/URL
gennames	/usr/bin/gennames	bos.perf.tools
gprof	/usr/bin/gprof	bos.adt.prof
iostat	/usr/bin/iostat	bos.acct
ipcs	/usr/bin/ipcs	bos.rte.control
ipfilter	/usr/bin/ipfilter	bos.perf.tools
ipreport	/usr/sbin/ipreport	bos.net.tcp.server
iptrace	/usr/sbin/iptrace	bos.net.tcp.server
jazizo (PTX)	/usr/bin/jazizo	perfmgr.analysis.jazizo
locktrace	/usr/bin/locktrace	bos.perf.tools
lslv	/usr/sbin/lslv	bos.rte.lvm
lspv	/usr/sbin/lspv	bos.rte.lvm
lsvg	/usr/sbin/lsvg	bos.rte.lvm
lvmstat	/usr/sbin/lvmstat	bos.rte.lvm
netpmon	/usr/bin/netpmon	bos.perf.tools
netstat	/usr/bin/netstat	bos.net.tcp.client
nfso	/usr/sbin/nfso	bos.net.nfs.client
nfsstat	/usr/sbin/nfsstat	bos.net.nfs.client
nice	/usr/bin/nice	bos.rte.control
no	/usr/sbin/no	bos.net.tcp.client
PDT	/usr/sbin/perf/diag_tool	bos.perf.diag_tool
perfpmr	-	ftp://software.ibm.com/
Perfstat API	-	bos.perf.libperfstat
PM API	-	bos.pmapi.lib
pprof	/usr/bin/pprof	bos.perf.tools
prof	/usr/bin/prof	bos.adt.prof
ps	/usr/bin/ps	bos.rte.control
renice	/usr/bin/renice	bos.rte.control

Command or tool	Full path name	Fileset name/URL
RMC	-	rsct.*
rmss	/usr/bin/rmss	bos.perf.tools
sar	/usr/sbin/sar	bos.acct
schedtune	/usr/samples/kernel/schedtune	bos.adt.samples
splat	-	ftp://software.ibm.com/
SPMI API	-	perfagent.tools, perfagent.server
stripnm	/usr/bin/stripnm	bos.perf.tools
svmon	/usr/bin/svmon	bos.perf.tools
tcpdump	/usr/sbin/tcpdump	bos.net.tcp.server
time	/usr/bin/time	bos.rte.misc_cmds
timex	/usr/bin/timex	bos.acct
tokstat	/usr/bin/tokstat	devices.common.IBM.tokenring.rte
topas	/usr/bin/topas	bos.perf.tools
tprof	/usr/bin/tprof	bos.perf.tools
trace	/usr/bin/trace	bos.sysmgt.trace
trcnm	/usr/bin/trcnm	bos.sysmgt.trace
trcrpt	/usr/bin/trcrpt	bos.sysmgt.trace
trpt	/usr/sbin/trpt	bos.net.tcp.server
truss	/usr/bin/truss	bos.sysmgt.serv_aid
vmstat	/usr/bin/vmstat	bos.acct
vmtune	/usr/samples/kernel/vmtune	bos.adt.samples
wlmmon	/usr/bin/wlmmon	perfagent.tools
wlmperf	/usr/bin/wlmperf	perfmgr.analysis.jazizo
wlmstat	/usr/sbin/wlmstat	bos.rte.control
xmperf (PTX)	/usr/bin/xmperf	perfmgr.network

Tools by resource matrix

This section contains a table of the AIX monitoring and tuning tools (Table B-2) and what system resources (CPU, memory, disk I/O, network I/O) they obtain statistics for. Tools that are used by **trace**, that post-process the **trace** output, or that are directly related to **trace** are listed in the Trace tools column. Tools that are useful for application development are listed in the Application development column.

Table B-2 Performance tools by resource matrix

Command	CPU	Memory	Disk I/O	Network I/O	Trace tools	Application Development
alstat	x					
atmstat				x		
bindintcpu	x					
bindprocessor	x					
curt	x				x	
emstat	x					
entstat				x		
estat				x		
fddistat				x		
fdpr						x
filemon			x		x	
fileplace			x			
genkex					x	
genkld					x	
genld					x	
gennames					x	
gprof	x					x
iostat	x		x			
ipcs		x				x
ipfilter				x		

Command	CPU	Memory	Disk I/O	Network I/O	Trace tools	Application Development
ipreport				x		
iptrace				x		
locktrace	x				x	
lslv			x			
lspv			x			
lsvg			x			
lvmstat			x			
netpmon	x			x	x	
netstat				x		
nfso				x		
nfsstat				x		
nice	x					
no				x		
PDT	x	x	x	x		
perfpmr	x	x	x	x	x	
Perfstat API	x	x	x	x		
PM API	x					x
pprof	x				x	
prof	x					x
ps	x	x				
PTX	x	x	x	x		x
renice	x					
RMC	x	x	x	x		
rmss		x				
sar	x	x	x	x		
schedtune	x	x				

Command	CPU	Memory	Disk I/O	Network I/O	Trace tools	Application Development
splat	x				x	x
SPMI API	x	x	x	x		
stripnm						
svmon		x				
tcpdump				x		
time	x					
timex	x					
tokstat				x		
topas	x	x	x	x		
tprof	x				x	x
trace	x	x	x	x	x	
trcnm					x	
trcrpt					x	
trpt				x		x
truss					x	
vmstat	x	x	x			
vmtune		x				
wlmmon	x	x	x	x		
wlmperf	x	x	x	x		
wlmstat	x	x	x	x		

The bos.perf package, part of the base operating system, contains the bos.perf.diag_tool fileset with the performance diagnostic tool. Performance PMR data collection scripts were removed from this fileset in AIX Version 4.3.3. They are available from the IBM support download Web site. In AIX 5L Version 5.1, the bos.perf.diag_tool fileset is part of base operating system.

Performance Toolbox

The Performance Toolbox is a Motif-based, AIX Licensed Program Product (LPP) that consolidates AIX performance tools in a toolbox framework. Users can easily access tools for system and network performance tuning, monitoring, and analysis. It consists of two major components: Performance Toolbox Manager and Performance Toolbox Agent. The Performance Toolbox Manager has three packages:

perfmgr.local This package contains the commands and utilities that allow monitoring of only the local system.

perfmgr.network This package contains the commands and utilities that allow monitoring of remote systems as well as the local system.

perfmgr.common This package contains the commands and utilities that are common between the network support and the local support.

The Performance Toolbox Agent has one package:

perfagent.server This package contains the performance agent component required by Performance Toolbox as well as some local AIX analysis and control tools.

The packaging of the older levels of the Performance Toolbox contained two system-level dependant filesets named perfagent.server and perfagent.tool, causing installation misunderstandings. To reduce confusion over operating system compatibility, the pieces that are required to be built with the AIX kernel have been moved into the perfagent.tools fileset. The agent is now, mainly, an interface routine to those pieces.

Starting with AIX Version 4.3.2, the perfagent.tools fileset is shipped with the base. For AIX Version 4.3.2 and above, the Performance Toolbox Agent will require perfagent.tools as a prerequisite. Therefore, the tools fileset must be installed first.

Table B-3 lists the various minimum fileset levels required with a particular AIX level.

Table B-3 Performance Toolbox releases

AIX version	Performance Agent	Performance Manager
AIX 4.1.5	perfagent 2.1.6.0	perfmgr 2.2.1.0 perfmgr.common 2.2.1.2 perfmgr.local 2.2.1.4 perfmgr.network 2.2.1.4

AIX version	Performance Agent	Performance Manager
AIX 4.2.1	perfagent 2.2.1.0	perfmgr 2.2.1.0 perfmgr.common 2.2.1.2 perfmgr.local 2.2.1.4 perfmgr.network 2.2.1.4
AIX 4.3.1	perfagent.tools 2.2.31.0	perfmgr 2.2.1.0 perfmgr.common 2.2.1.2 perfmgr.local 2.2.1.4 perfmgr.network 2.2.1.4
AIX 4.3.2	perfagent.tools 2.2.32.0 perfagent.server 2.2.32.0	perfmgr 2.2.1.0 perfmgr.common 2.2.1.2 perfmgr.local 2.2.1.4 perfmgr.network 2.2.1.4
AIX 4.3.3	perfagent.tools 2.2.33.82 perfagent.server 2.2.32.10	perfmgr 2.2.1.10 perfmgr.common 2.2.1.10 perfmgr.local 2.2.1.10 perfmgr.network 2.2.1.10
AIX 5.1.0	perfagent.tools 5.1.0.27 perfagent.server 3.0.0.0	perfmgr.common 3.0.0.0 perfmgr.network 3.0.0.0

As listed, the Performance Manager releases remain the same throughout AIX Version 4.1.5 to 4.3.3. However, you should choose the correct version of the Agent component, because it will only work properly when installed on the correct level of AIX.

Before you install Performance Toolbox software, you have to determine the AIX and the maintenance level of your system. To determine the AIX and maintenance level of the system, enter the following command:

```
# oslevel
5.1.0.0
```

The current maintenance level of the system shown in this example is 5.1.0.0. To list the names of known maintenance levels, use the following command:

```
# oslevel -q
Known Maintenance Levels
------------------------
5.1.0.0
5.0.0.0
```

To determine the filesets at levels later than the current AIX maintenance level, enter:

```
# oslevel -g|more
```

```
Fileset                                 Actual Level        Maintenance Level
-----------------------------------------------------------------------------
IMNSearch.bld.DBCS                      2.3.1.15            2.3.1.0
IMNSearch.bld.SBCS                      2.3.1.15            2.3.1.0
IMNSearch.rte.com                       2.3.1.15            2.3.1.0
IMNSearch.rte.httpdlite                 2.0.0.15            2.0.0.2
Java130.rte.bin                         1.3.0.14            1.3.0.5
Java130.rte.lib                         1.3.0.14            1.3.0.5
X11.Dt.ToolTalk                         5.1.0.15            5.1.0.0
.....
.....
```

Another command that can be used to determine the software levels installed is the **instfix** command. To list the names of installed maintenance levels on another server with AIX5L Version 5.1 installed, use the following command:

```
# instfix -i |grep ML
    All filesets for 5.0.0.0_AIX_ML were found.
    All filesets for 5.1.0.0_AIX_ML were found.
    All filesets for 5.1.0.0_AIX_ML were found.
    All filesets for 5100-01_AIX_ML were found.
    All filesets for 5100-02_AIX_ML were found.
```

The **instfix** command allows you to install a fix or set of fixes without knowing any information other than the Authorized Program Analysis Report (APAR) number. The **instfix** command also allows you to determine if a fix is installed on your system.

The **oslevel** command reports the level of the operating system using a subset of all filesets installed on you system. These filesets include the Base Operating System (BOS), and the device drivers and base printers. The **oslevel** command also prints information about maintenance levels, including which filesets are not at a specified maintenance level.

To check if any of the Performance Toolbox packages are installed, use the following command:

```
# lslpp -l perf*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos                5.1.0.25            5.1.0.0
  perfagent.server           3.0.0.0  COMMITTED  Performance Agent Daemons &
                                                 Utilities
  perfmgr.common             3.0.0.0  COMMITTED  Performance Toolbox Manager -
                                                 Common Support
  perfmgr.network            3.0.0.0  COMMITTED  Performance Toolbox Manager -
                                                 Network Support
```

```
Path: /etc/objrepos
  perfagent.server           3.0.0.0  COMMITTED  Performance Agent Daemons &
                                                 Utilities
```

You can receive similar information using the **smitty list_software** command. The output is shown in Figure B-1.

```
                                COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

[TOP]
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  perfagent.html.en_US.usergd
                             5.1.0.0    C     F    Performance Toolbox Guides -
U.
                                                   S. English
  perfagent.server           3.0.0.0    C     F    Performance Agent Daemons &
                                                   Utilities
  perfagent.tools           5.1.0.25    C     F    Local Performance Analysis &
                                                   Control Commands
  perfmgr.common             3.0.0.0    C     F    Performance Toolbox Manager -
                                                   Common Support
[MORE...18]

F1=Help             F2=Refresh          F3=Cancel           F6=Command
F8=Image            F9=Shell            F10=Exit            /=Find
n=Find Next
```

Figure B-1 smitty list_software output

If you are missing some filesets and have media containing the Performance Toolbox software, you can check what it contains. You can use either the **installp** command or **smitty**. The Web-based System Manager is an additional tool for system administration and is the strategic interface.

The output from the **installp** command appears as follows:

```
# installp -ld . -I
  Fileset Name                 Level                      I/U Q Content
  ============================================================================
  perfagent.server             3.0.0.0                     I  N usr,root
#   Performance Agent Daemons & Utilities

  perfmgr.common               3.0.0.0                     I  N usr
#   Performance Toolbox Manager - Common Support

  perfmgr.network              3.0.0.0                     I  N usr
#   Performance Toolbox Manager - Network Support
```

This indicates that the media contains both components: The Performance Toolbox Manager and the Performance Toolbox Agent.

The next step is to install the Performance Toolbox. The most direct way to install software is with the **smitty install_all** command, as shown in Figure B-2.

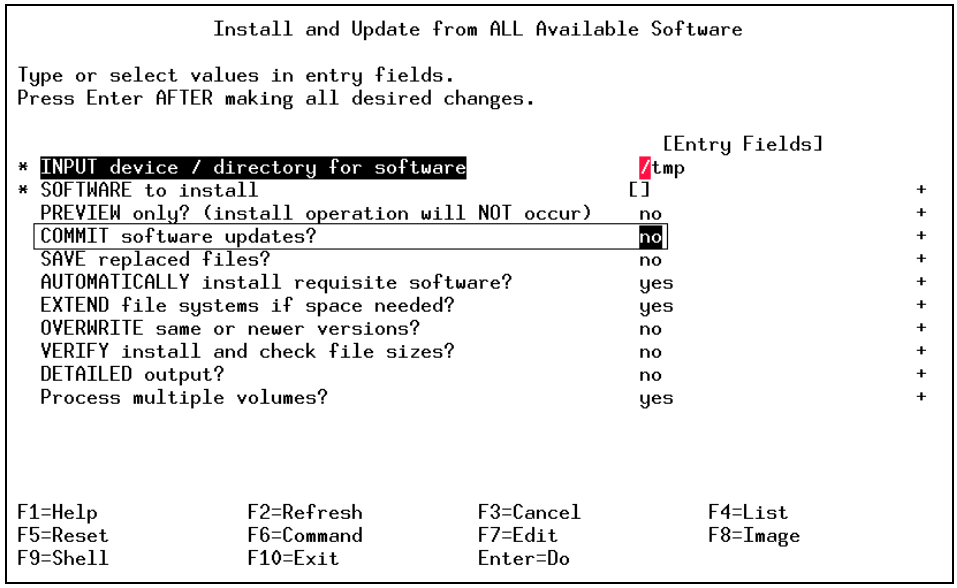


Figure B-2 smitty install_all

Do not commit software until you are sure that the installation process does not impact the system. After installation, check the $HOME/smit.log file for errors and run the **lppchk** command to verify a successful installation process. If everything is normal, you can commit your installation with the **installp -c all** command. Or use SMIT as shown below:

```
# smitty commit
```

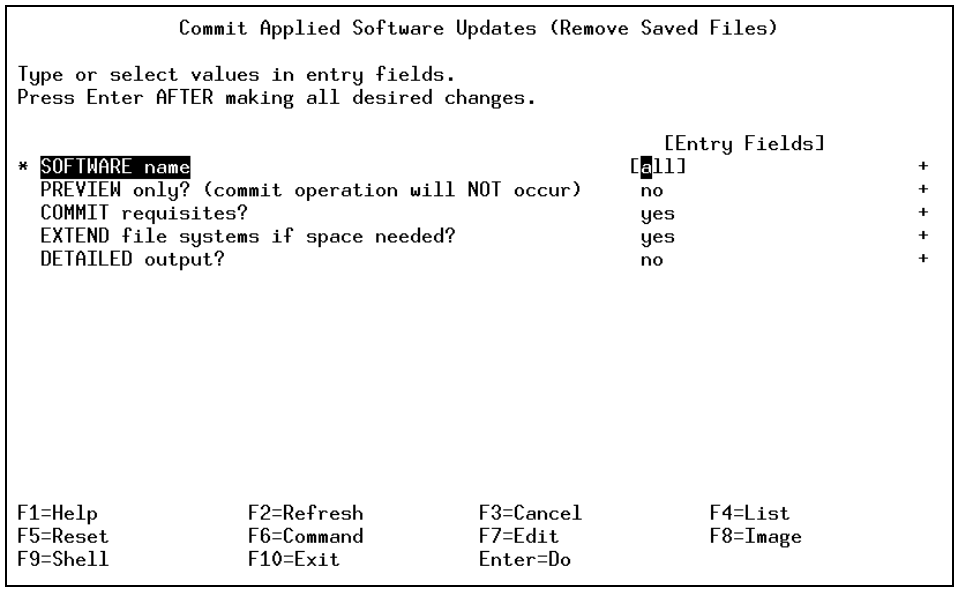


Figure B-3 Commit software updates

Clean up any failed installation with the **installp -C** command.

For example, to check that you have the filesets installed for both the manager and agent part, perform the following steps (a server with AIX Version 4.3.3 was used).

For the agent, enter the following command:

```
# lslpp -l perfagent.*
  Fileset                      Level  State      Description
----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  perfagent.html.en_US.usergd
                             4.3.0.0  COMMITTED  Performance Toolbox Guides -
                                                 U. S. English
  perfagent.server          2.2.32.3  APPLIED    Performance Agent Daemons &
                                                 Utilities
  perfagent.tools           2.2.34.0  COMMITTED  Local Performance Analysis&
                                                 Control Commands

Path: /etc/objrepos
  perfagent.server          2.2.30.0  COMMITTED  Performance Agent Daemons &
                                                 Utilities
```

For the manager, enter the following command:

```
# lslpp -l perfmgr.*
  Fileset                      Level  State      Description
----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  perfmgr.common             2.2.1.5  APPLIED    Performance Toolbox Manager
                                                 Common Support
  perfmgr.local              2.2.1.7  APPLIED    Performance Toolbox Manager
                                                 Local Support
  perfmgr.network            2.2.1.7  APPLIED    Performance Toolbox Manager
                                                 Network Support
```

Examine what is inside the filesets that you installed. This completes the installation:

```
# lslpp -f perfmgr.local
  Fileset               File
----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  perfmgr.local 2.2.1.0
                        /usr/lpp/perfmgr/local/bin
                        /usr/lpp/perfmgr/local/bin/3dmon
                        /usr/lpp/perfmgr
                        /usr/lpp/perfmgr/local/bin/exmon
                        /usr/lpp/perfmgr/local
                        /usr/lpp/perfmgr/local/bin/xmperf
                        /usr/lpp/perfmgr/README.perfmgr.local
                        /usr/lpp/perfmgr/local/bin/ptxrlog
```

Another tool used by IBM support personnel is **perfpmr**, a collection facility that uses standard AIX performance commands. Go to the following site and follow the readme file for further instructions on how to install this utility:

ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr/perf51/

Command summary

The following section provides a list of the key commands discussed in this appendix.

The installp command

The **installp** command is a very useful and powerful software installation tool.

To install with apply only or with apply and commit:

```
installp [ -a | -ac [ -N ] ] [ -eLogFile ] [ -V Number ] [ -dDevice ] [ -b ]
[ -S ] [ -B ] [ -D ] [ -I ] [ -p ] [ -Q ] [ -q ] [ -v ] [ -X ]
[-F | -g ] [ -O { [ r ] [ s ] [ u ] } ] [ -tSaveDirectory ] [ -w ]
[ -zBlockSize ] { FilesetName [ Level ]... | -f ListFile | all }
```

To commit applied updates:

```
installp -c [ -eLogFile ] [ -VNumber ] [ -b ] [ -g ] [ -p ] [ -v ]
[ -X ] [ -O { [ r ] [ s ] [ u ] } ] [ -w ] { FilesetName [ Level ]... | -f
ListFile | all }
```

To reject applied updates:

```
installp -r [ -eLogFile ] [ -VNumber ] [ -b ] [ -g ] [ -p ] [ -v ] [ -X ]
[ -O { [ r ] [ s ] [ u ] } ] [ -w ] { FilesetName [ Level ]... |
-f ListFile }
```

To uninstall (remove) installed software:

```
installp -u [ -eLogFile ] [ -VNumber ] [ -b ] [ -g ] [ -p ] [ -v ] [ -X ]
[ -O { [ r ] [ s ] [ u ] } ] [ -w ] { FilesetName [ Level ]... |
-f ListFile }
```

To clean up a failed installation:

```
installp -C [ -b ] [ -eLogFile ]
```

To list all installable software on a selected media:

```
installp { -l | -L } [ -eLogFile ] [ -dDevice ] [ -B ] [ -I ] [ -q ]
[ -zBlockSize ] [ -O { [ s ] [ u ] } ]
```

To list all customer-reported problems fixed with software or display all supplemental information:

```
installp { -A | -i } [ -eLogFile ] [ -dDevice ] [ -B ] [ -I ] [ -q ]
[ -z BlockSize ] [ -O { [ s ] [ u ] } ] { FilesetName [ Level ]... | -f
ListFile | all }
```

To list installed updates that are applied but not committed:

```
installp -s [ -eLogFile ] [ -O { [ r ] [ s ] [ u ] } ] [ -w ]
{ FilesetName [ Level ]... | -fListFile | all }
```

Table B-4 provides a general summary of some useful **installp** flags.

Table B-4 General installp summary

Flag	Description
-ac	Commits

Flag	Description
-g	Includes requisites
-N	Overrides saving of existing files
-q	Quiet mode
-w	Does not place a wildcard at end of fileset name
-X	Attempts to expand file system size if needed
-d	Inputs device
-l	Lists installable filesets
-c	Commits an applied fileset
-C	Cleans up after an failed installation
-u	Uninstalls
-r	Rejects an applied fileset
-p	Previews installation
-e	Defines an installation log
-F	Forces overwrite of same or newer version

The lslpp command

The **lslpp** command displays information about installed filesets or fileset updates. The command has the following syntax:

```
lslpp { -f | -h | -i | -L } ] [ -a ] [ FilesetName ... | FixID ... | all ]
```

Table B-5 provides a general summary of some useful **lslpp** flags.

Table B-5 Commonly used flags of the lslpp command

Flag	Description
-a	Displays all the information about filesets specified when combined with other flags.
-f	Displays the names of the files added to the system during installation of the specified fileset. This flag cannot be specified with the -a flag.
-h	Displays the installation and update history information for the specified fileset.
-i	Displays the product information for the specified fileset.

Flag	Description
-L	Displays the name, most recent level, state, and description of the specified fileset. Part of the information (usr, root, and share) is consolidated into the same listing.
-w	Lists fileset that owns this file.

The lppchk command

The **lppchk** command verifies files of an installable software product. The command has the following syntax:

```
lppchk { -c | -f | -l | -v } [ -O { [ r ] [ s ] [ u ] } ]
[ ProductName [ FileList ... ] ]
```

Table B-6 provides a general summary of some useful **lppchk** flags.

Table B-6 Commonly used flags of the lppchk command

Flag	Description
-c	Performs a checksum operation on the FileList items and verifies that the checksum and the file size are consistent with the SWVPD database.
-f	Checks that the FileList items are present and the file size matches the SWVPD database.
-l	Verifies symbolic links for files as specified in the SWVPD database.
-O {[r] [s] [u]}	Verifies the specified parts of the program. The flags specify the following parts: root, share, usr.

Quiz

The following assessment questions help verify your understanding of the topics discussed in this appendix.

1. Which of the following commands should be used to obtain information about the level of the operating system and conformance with maintenance levels?

   A. **lslpp**

   B. **uname**

   C. **oslevel**

   D. **rpcinfo**

2. Which of the following commands should be used to obtain information about the installed software filesets?

   A. **lslpp**

   B. **uname**

   C. **oslevel**

   D. **rpcinfo**

3. Which of the following commands should be used to obtain information about the fixes available on some media?

   A. **installp -Ad . all**

   B. **/usr/sbin/inutoc .**

   C. **lsresource -dl rmt0**

   D. **/usr/sbin/lsattr -Dl rmt0**

4. Which of the following commands should be used to commit applied filesets?

   A. **/usr/sbin/lsattr**

   B. **/usr/sbin/inutoc**

   C. **/usr/sbin/installp**

   D. **/usr/sbin/lsresource**

5. An AIX Version 4.3 system must be evaluated and tuned. In order to use a set of performance tools (for example, **filemon**, **svmon**, and **tprof**) which of the following packages should exist on the system in question?

   A. Java.adt

   B. bos.perf

   C. bos.adt.utils

   D. perfagent.tools

6. Which of the following packages once contained the Performance PMR Data Collection scripts (removed in AIX Version 4.3.3, although they are available for download from the perfpmr site)?

   A. bos.perf

   B. Java.adt

   C. bos.adt.utils

   D. perfagent.tools

7. Which of the following resources will provide the latest available performance data capture tools?

   A. The system's software depositories

   B. The IBM support download Web site

   C. The standard documentation included with the system

   D. The software distribution media included with the system

8. Which of the following filesets is used to install **tprof**?

   A. bos.acct

   B. bos.perf.pmr

   C. bos.rte.control

   D. perfagent.tools

9. The components **xmperf**, **3dmon**, and **3dplay** are part of the Performance Toolbox contained in which of the following packages?

   A. perfmgr

   B. perfagent.server

   C. perfagent.tools

   D. perfagent.client

Answers

The following are the preferred answers to the questions provided in this section.

1. C
2. A
3. A
4. C
5. D
6. A
7. B
8. D
9. A

Exercises

The following exercises provide sample topics for self study. They will help ensure comprehension of this appendix.

1. Use the **lslpp** command to list installed filesets.
2. Use the **lslpp** command to find out which fileset is used to package a given command.
3. Use the **lslpp** command to display state, description, and all updates of the different filesets.
4. Use the **oslevel** command to determine the filesets at levels later than the current AIX maintenance level.

Abbreviations and acronyms

ABI	Application Binary Interface
AC	Alternating Current
ACL	Access Control List
ADSM	ADSTAR Distributed Storage Manager
ADSTAR	Advanced Storage and Retrieval
AFPA	Adaptive Fast Path Architecture
AFS	Andrew File System
AH	Authentication Header
AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
APAR	Authorized Program Analysis Report
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCI	Accelerated Strategic Computing Initiative
ASCII	American National Standards Code for Information Interchange
ASR	Address Space Register
ATM	Asynchronous Transfer Mode
AuditRM	Audit Log Resource Manager
AUI	Attached Unit Interface
AWT	Abstract Window Toolkit
BCT	Branch on CounT
BFF	Backup File Format
BI	Business Intelligence
BIND	Berkeley Internet Name Daemon
BIST	Built-In Self-Test
BLAS	Basic Linear Algebra Subprograms
BLOB	Binary Large Object
BLV	Boot Logical Volume
BOOTP	Boot Protocol
BOS	Base Operating System
BSC	Binary Synchronous Communications
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CATE	Certified Advanced Technical Expert
CATIA	Computer-Graphics Aided Three-Dimensional Interactive Application
CCM	Common Character Mode
CD	Compact Disk
CDE	Common Desktop Environment
CDLI	Common Data Link Interface
CD-R	CD Recordable
CD-ROM	Compact Disk-Read Only Memory
CE	Customer Engineer
CEC	Central Electronics Complex
CFD	Computational Fluid Dynamics
CGE	Common Graphics Environment
CHRP	Common Hardware Reference Platform

CISPR	International Special Committee on Radio Interference
CLIO/S	Client Input/Output Sockets
CLVM	Concurrent LVM
CMOS	Complimentary Metal-Oxide Semiconductor
CMP	Certificate Management Protocol
COFF	Common Object File Format
COLD	Computer Output to Laser Disk
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSID	Character Set ID
CSR	Customer Service Representative
CSS	Communication Subsystems Support
CSU	Customer Set-Up
CWS	Control Workstation
DAD	Duplicate Address Detection
DAS	Dual Attach Station
DASD	Direct Access Storage Device
DAT	Digital Audio Tape
DBCS	Double Byte Character Set
DBE	Double Buffer Extension
DC	Direct Current
DCE	Distributed Computing Environment
DDC	Display Data Channel
DDS	Digital Data Storage
DE	Dual-Ended
DES	Data Encryption Standard
DFL	Divide Float
DFP	Dynamic Feedback Protocol
DFS	Distributed File System
DHCP	Dynamic Host Configuration Protocol
DIMM	Dual In-Line Memory Module
DIP	Direct Insertion Probe
DIT	Directory Information Tree
DIVA	Digital Inquiry Voice Answer
DLT	Digital Linear Tape
DMA	Direct Memory Access
DMT	Directory Management Tool
DN	Distinguished Name
DNS	Domain Naming System
DOE	Department of Energy
DOI	Domain of Interpretation
DOS	Disk Operating System
DPCL	Dynamic Probe Class Library
DRAM	Dynamic Random Access Memory
DS	Differentiated Service
DSA	Dynamic Segment Allocation
DSE	Diagnostic System Exerciser
DSMIT	Distributed SMIT
DSU	Data Service Unit
DTE	Data Terminating Equipment
DW	Data Warehouse
EA	Effective Address
EC	Engineering Change
ECC	Error Checking and Correcting
EEPROM	Electrically Erasable Programmable Read Only Memory
EFI	Extensible Firmware Interface
EHD	Extended Hardware Drivers
EIA	Electronic Industries Association
EISA	Extended Industry Standard Architecture

ELA	Error Log Analysis
ELF	Executable and Linking Format
EMU	European Monetary Union
EOF	End of File
EPOW	Environmental and Power Warning
ERRM	Event Response resource manager
ESID	Effective Segment ID
ESP	Encapsulating Security Payload
ESSL	Engineering and Scientific Subroutine Library
ETML	Extract, Transformation, Movement, and Loading
F/C	Feature Code
F/W	Fast and Wide
FC	Fibre Channel
FCAL	Fibre Channel Arbitrated Loop
FCC	Federal Communication Commission
FCP	Fibre Channel Protocol
FDDI	Fiber Distributed Data Interface
FDPR	Feedback Directed Program Restructuring
FDX	Full Duplex
FIFO	First In/First Out
FLASH EPROM	Flash Erasable Programmable Read-Only Memory
FLIH	First Level Interrupt Handler
FMA	Floating point Multiply Add operation
FPR	Floating Point Register
FPU	Floating Point Unit
FRCA	Fast Response Cache Architecture
FRU	Field Replaceable Unit
FSRM	File System Resource Manager
FTP	File Transfer Protocol
FTP	File Transfer Protocol
GAI	Graphic Adapter Interface
GAMESS	General Atomic and Molecular Electronic Structure System
GPFS	General Parallel File System
GPR	General-Purpose Register
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HACMP	High Availability Cluster Multi Processing
HACWS	High Availability Control Workstation
HCON	IBM AIX Host Connection Program/6000
HDX	Half Duplex
HFT	High Function Terminal
HIPPI	High Performance Parallel Interface
HiPS	High Performance Switch
HiPS LC-8	Low-Cost Eight-Port High Performance Switch
HMC	Hardware Management Console
HostRM	Host Resource Manager
HP	Hewlett-Packard
HPF	High Performance FORTRAN
HPSSDL	High Performance Supercomputer Systems Development Laboratory
HP-UX	Hewlett-Packard UNIX
HTML	Hyper-text Markup Language
HTTP	Hypertext Transfer Protocol
Hz	Hertz

I/O	Input/Output
I^2C	Inter Integrated-Circuit Communications
IAR	Instruction Address Register
IBM	International Business Machines
ICCCM	Inter-Client Communications Conventions Manual
ICE	Inter-Client Exchange
ICElib	Inter-Client Exchange library
ICMP	Internet Control Message Protocol
ID	Identification
IDE	Integrated Device Electronics
IDS	Intelligent Decision Server
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IHV	Independent Hardware Vendor
IIOP	Internet Inter-ORB Protocol
IJG	Independent JPEG Group
IKE	Internet Key Exchange
ILS	International Language Support
IM	Input Method
INRIA	Institut National de Recherche en Informatique et en Automatique
IP	Internetwork Protocol (OSI)
IPL	Initial Program Load
IPSec	IP Security
IrDA	Infrared Data Association (which sets standards for infrared support including protocols for data interchange)
IRQ	Interrupt Request
IS	Integrated Service
ISA	Industry Standard Architecture, Instruction Set Architecture
ISAKMP	Internet Security Association Management Protocol
ISB	Intermediate Switch Board
ISDN	Integrated-Services Digital Network
ISMP	InstallSheild Multi-Platform
ISNO	Interface Specific Network Options
ISO	International Organization for Standardization
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
JBOD	Just a Bunch of Disks
JDBC	Java Database Connectivity
JFC	Java Foundation Classes
JFS	Journaled File System
JTAG	Joint Test Action Group
KDC	Key Distribution Center
L1	Level 1
L2	Level 2
L2	Level 2
LAN	Local Area Network
LANE	Local Area Network Emulation
LAPI	Low-Level Application Programming Interface
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Directory Interchange Format
LED	Light Emitting Diode
LFD	Load Float Double
LFT	Low Function Terminal

LID	Load ID
LLNL	Lawrence Livermore National Laboratory
LP	Logical Partition
LP64	Long-Pointer 64
LPI	Lines Per Inch
LPP	Licensed Program Product
LPR/LPD	Line Printer/Line Printer Daemon
LRU	Least Recently Used
LTG	Logical Track Group
LV	Logical Volume
LVCB	Logical Volume Control Block
LVD	Low Voltage Differential
LVM	Logical Volume Manager
MAP	Maintenance Analysis Procedure
MASS	Mathematical Acceleration Subsystem
MAU	Multiple Access Unit
MBCS	Multi-Byte Character Support
Mbps	Megabits Per Second
MBps	Megabytes Per Second
MCA	Micro Channel Architecture
MCAD	Mechanical Computer-Aided Design
MDI	Media Dependent Interface
MES	Miscellaneous Equipment Specification
MFLOPS	Million of FLoating point Operations Per Second
MII	Media Independent Interface
MIP	Mixed-Integer Programming
MLR1	Multi-Channel Linear Recording 1
MMF	Multi-Mode Fibre
MODS	Memory Overlay Detection Subsystem
MP	Multiprocessor
MPC-3	Multimedia PC-3
MPI	Message Passing Interface
MPOA	Multiprotocol over ATM
MPP	Massively Parallel Processing
MPS	Mathematical Programming System
MST	Machine State
MTU	Maximum Transmission Unit
MWCC	Mirror Write Consistency Check
MX	Mezzanine Bus
NBC	Network Buffer Cache
NCP	Network Control Point
ND	Neighbor Discovery
NDP	Neighbor Discovery Protocol
NFB	No Frame Buffer
NFS	Network File System
NHRP	Next Hop Resolution Protocol
NIM	Network Installation Management
NIS	Network Information System
NL	National Language
NLS	National Language Support
NT-1	Network Terminator-1
NTF	No Trouble Found
NTP	Network Time Protocol
NUMA	Non-Uniform Memory Access
NUS	Numerical Aerodynamic Simulation
NVRAM	Non-Volatile Random Access Memory
NWP	Numerical Weather Prediction
OACK	Option Acknowledgment
OCS	Online Customer Support
ODBC	Open DataBase Connectivity
ODM	Object Data Manager

OEM	Original Equipment Manufacturer
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
ONC+	Open Network Computing
OOUI	Object-Oriented User Interface
OSF	Open Software Foundation, Inc.
OSL	Optimization Subroutine Library
OSLp	Parallel Optimization Subroutine Library
P2SC	POWER2 Single/Super Chip
PAM	Pluggable Authentication Mechanism
PAP	Privileged Access Password
PBLAS	Parallel Basic Linear Algebra Subprograms
PCI	Peripheral Component Interconnect
PDT	Paging Device Table
PDU	Power Distribution Unit
PE	Parallel Environment
PEDB	Parallel Environment Debugging
PEX	PHIGS Extension to X
PFS	Perfect Forward Security
PGID	Process Group ID
PHB	Processor Host Bridges
PHY	Physical Layer
PID	Process ID
PID	Process ID
PIOFS	Parallel Input Output File System
PKR	Protection Key Registers
PMTU	Path MTU
POE	Parallel Operating Environment
POP	Power-On Password
POSIX	Portable Operating Interface for Computing Environments
POST	Power-On Self-test
POWER	Performance Optimization with Enhanced Risc (Architecture)
PPC	PowerPC
PPM	Piecewise Parabolic Method
PPP	Point-to-Point Protocol
PREP	PowerPC Reference Platform
PSE	Portable Streams Environment
PSSP	Parallel System Support Program
PTF	Program Temporary Fix
PTPE	Performance Toolbox Parallel Extensions
PTX	Performance Toolbox
PV	Physical Volume
PVC	Permanent Virtual Circuit
PVID	Physical Volume Identifier
QMF	Query Management Facility
QoS	Quality of Service
QP	Quadratic Programming
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAN	Remote Asynchronous Node
RAS	Reliability, Availability, and Serviceability
RDB	Relational DataBase
RDBMS	Relational Database Management System
RDISC	ICMP Router Discovery
RDN	Relative Distinguished Name

RDP	Router Discovery Protocol
RFC	Request for Comments
RIO	Remote I/O
RIP	Routing Information Protocol
RIPL	Remote Initial Program Load
RISC	Reduced Instruction-Set Computer
RMC	Resource Monitoring and Control
ROLTP	Relative Online Transaction Processing
RPA	RS/6000 Platform Architecture
RPC	Remote Procedure Call
RPL	Remote Program Loader
RPM	Redhat Package Manager
RSC	RISC Single Chip
RSCT	Reliable Scalable Cluster Technology
RSE	Register Stack Engine
RSVP	Resource Reservation Protocol
RTC	Real-Time Clock
RVSD	Recoverable Virtual Shared Disk
SA	Secure Association
SACK	Selective Acknowledgments
SAN	Storage Area Network
SAR	Solutions Assurance Review
SAS	Single Attach Station
SBCS	Single-Byte Character Support
ScaLAPACK	Scalable Linear Algebra Package
SCB	Segment Control Block
SCSI	Small Computer System Interface
SCSI-SE	SCSI-Single Ended
SDLC	Synchronous Data Link Control
SDR	System Data Repository
SDRAM	Synchronous Dynamic Random Access Memory
SE	Single Ended
SEPBU	Scalable Electrical Power Base Unit
SGI	Silicon Graphics Incorporated
SGID	Set Group ID
SHLAP	Shared Library Assistant Process
SID	Segment ID
SIT	Simple Internet Transition
SKIP	Simple Key Management for IP
SLB	Segment Lookaside Buffer
SLIH	Second Level Interrupt Handler
SLIP	Serial Line Internet Protocol
SLR1	Single-Channel Linear Recording 1
SM	Session Management
SMB	Server Message Block
SMIT	System Management Interface Tool
SMP	Symmetric Multiprocessor
SMS	System Management Services
SNG	Secured Network Gateway
SOI	Silicon-on-Insulator
SP	IBM RS/6000 Scalable POWER parallel Systems
SP	Service Processor
SPCN	System Power Control Network
SPEC	System Performance Evaluation Cooperative
SPI	Security Parameter Index

SPM	System Performance Measurement
SPOT	Shared Product Object Tree
SPS	SP Switch
SPS-8	Eight-Port SP Switch
SRC	System Resource Controller
SRN	Service Request Number
SSA	Serial Storage Architecture
SSC	System Support Controller
SSL	Secure Socket Layer
STFDU	Store Float Double with Update
STP	Shielded Twisted Pair
SUID	Set User ID
SUP	Software Update Protocol
SVC	Switch Virtual Circuit
SVC	Supervisor or System Call
SWVPD	Software Vital Product Data
SYNC	Synchronization
TCE	Translate Control Entry
Tcl	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TCQ	Tagged Command Queuing
TGT	Ticket Granting Ticket
TLB	Translation Lookaside Buffer
TOS	Type Of Service
TPC	Transaction Processing Council
TPP	Toward Peak Performance
TSE	Text Search Engine
TSE	Text Search Engine
TTL	Time To Live
UCS	Universal Coded Character Set
UDB EEE	Universal Database and Enterprise Extended Edition
UDI	Uniform Device Interface
UIL	User Interface Language
ULS	Universal Language Support
UP	Uniprocessor
USB	Universal Serial Bus
USLA	User-Space Loader Assistant
UTF	UCS Transformation Format
UTM	Uniform Transfer Model
UTP	Unshielded Twisted Pair
UUCP	UNIX-to-UNIX Communication Protocol
VESA	Video Electronics Standards Association
VFB	Virtual Frame Buffer
VG	Volume Group
VGDA	Volume Group Descriptor Area
VGSA	Volume Group Status Area
VHDCI	Very High Density Cable Interconnect
VLAN	Virtual Local Area Network
VMM	Virtual Memory Manager
VP	Virtual Processor
VPD	Vital Product Data
VPN	Virtual Private Network
VSD	Virtual Shared Disk
VSM	Visual System Manager
VSS	Versatile Storage Server
VT	Visualization Tool
WAN	Wide Area Network
WLM	Workload Manager
WTE	Web Traffic Express
XCOFF	Extended Common Object File Format
XIE	X Image Extension
XIM	X Input Method
XKB	X Keyboard Extension

XLF	XL Fortran
XOM	X Output Method
XPM	X Pixmap
XSSO	Open Single Sign-on Service
XTF	Extended Distance Feature
XVFB	X Virtual Frame Buffer

Related publications

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

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 331.

- *AIX 5L Differences Guide*, SG24-5765
- *AIX 5L Performance Tools Handbook*, SG24-6039
- *AIX 5L Workload Manager (WLM)*, SG24-5977
- *AIX Logical Volume Manager From A to Z: Introduction and Concepts*, SG24-5432
- *IBM @server Certification Study Guide - AIX Communications*, SG24-6186
- *IBM @server Certification Study Guide - AIX Installation and System Recovery*, SG24-6183
- *IBM @server Certification Study Guide - AIX Problem Determination Tools and Techniques*, SG24-6185
- *IBM @server Certification Study Guide - pSeries AIX System Administration*, SG24-6191
- *IBM @server Certification Study Guide - pSeries AIX System Support*, SG24-6199
- *IBM @server Certification Study Guide - pSeries HACMP for AIX*, SG24-6187
- *IBM @server Certification Study Guide - RS/6000 SP*, SG24-5348
- *Managing AIX Server Farms*, SG24-6606
- *Problem Solving and Troubleshooting in AIX 5L*, SG24-5496
- *RS/6000 Performance Tools in Focus*, SG24-4989
- *TCP/IP Tutorial and Technical Overview*, GG24-3376

Other resources

These publications are also relevant as further information sources:

- *AIX Version 4 System Management Guide: Operating System and Devices*, SC23-2525
- *AIX Version 4.3 System Management Concepts: Operating System and Devices*, SC23-4311
- *RS/6000 and pSeries PCI Adapter Placement Reference,* SA38-0538
- *SSA Adapters: User's Guide and Maintenance Information*, SA33-3272
- The following types of documentation are located through the Internet at the following URL:

  `http://www-1.ibm.com/servers/eserver/pseries/library`

  - User guides
  - System management guides
  - Application programmer guides
  - All commands reference volumes
  - Files reference
  - Technical reference volumes used by application programmers

Referenced Web sites

These Web sites are also relevant as further information sources:

- IBM certification tests information

  `http://www.ibm.com/certify/tests/info.shtml`
- IBM eServer pSeries support

  `http://techsupport.services.ibm.com/server/support?view=pSeries`
- IBM hardware documentation

  `http://www.ibm.com/servers/eserver/pseries/library/hardware_docs/index.html`
- IBM Professional Certification Program

  `http://www.ibm.com/certify`
- IBM TotalStorage

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

- Open Group Technical Standard, Protocols for Interworking XNFS Version 3W

  http://www.opengroup.org/onlinepubs/9629799/toc.htm

- **perfpmr** installation readme file

  ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr/perf51/

- UNIX servers (pSeries) information

  http://www-132.ibm.com/content/home/store_IBMPublicUSA/en_US/eServer/pSeries/pSeries.html

How to get IBM Redbooks

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

ibm.com/redbooks

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

IBM Redbooks collections

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

Index

D

M

N

O

P

Q

R

S

T

U

V

W

X

IBM eserver Certification Study Guide - AIX 5L Performance and System Tuning

Developed specifically for the purpose of preparing for AIX certification

Makes an excellent companion to classroom education

For experienced AIX professionals

This IBM Redbook is designed as a study guide for professionals wishing to prepare for the AIX 5L Performance and System Tuning certification exam as a selected course of study in order to achieve the IBM eserver Certified Advanced Technical Expert - pSeries and AIX 5L certification.

This IBM Redbook is designed to provide a combination of theory and practical experience needed for a general understanding of the subject matter. It also provides sample questions that will help with the evaluation of personal progress and provide familiarity with the types of questions that will be encountered on the exam.

This publication does not replace practical experience, nor is it designed to be a stand-alone guide for any subject. Instead, it is an effective tool that, when combined with education activities and experience, can be a very useful preparation guide for the exam. Whether you are planning to take the AIX 5L Performance and System Tuning certification exam, or if you just want to validate your AIX skills, this redbook is for you.

This publication was updated to include the new content included in Test 234, which is based on AIX 5L Version 5.1.

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

SG24-6184-01 ISBN 0738427012