IBM Certified™
Specialist

IBM Certification Study Guide AIX Version 4.3 Performance and System Tuning

Thomas C. Cederlöf
André de Klerk
Thomas Herlin
Tomasz Ostaszewski

Redbooks

ibm.com/redbooks

SG24-6184-00

International Technical Support Organization

IBM Certification Study Guide
AIX Version 4.3
Performance and System Tuning

October 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 265.

First Edition (October 2000)

This edition applies to AIX Version 4.3 (5765-C34) and subsequent releases running on an RS/6000 server.

This document created or updated on July 21, 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures ix

Tables xi

Preface xiii
The team that wrote this redbook xiv
Comments welcome xv

Chapter 1. Certification overview 17
1.1 IBM Certified Advanced Technical Expert - RS/6000 AIX 17
1.1.1 Required prerequisite 17
1.1.2 Recommended prerequisite 17
1.1.3 Registration for the certification exam 17
1.1.4 Core requirement (select three of the following tests) 18
1.2 Certification education courses 31
1.3 Education on CD: IBM AIX Essentials 34

Chapter 2. Installing the performance tools 37
2.1 Commands 43
2.1.1 installp 43
2.1.2 lslpp 44
2.1.3 lppchk 45
2.2 References 46
2.3 Quiz 46
2.3.1 Answers 46
2.4 Exercises 46

Chapter 3. Performance tuning - getting started 47
3.1 Introduction to concepts 47
3.2 CPU bound 49
3.3 Memory bound 54
3.4 Disk Bound 63
3.5 Network bound 69
3.6 Summary 73
3.7 Quiz 74
3.7.1 Quiz answers 74

Chapter 4. CPU performance tools 75
4.1 The AIX scheduler 75
4.1.1 Priority calculation on AIX versions prior to 4.3.2 76
4.1.2 Priority calculation on AIX version 4.3.2 and later 79
4.2 Multiple run queues with load balancing in AIX Version 4.3.3 81

4.2.1 Load balancing 83
4.3 The schedtune command 84
4.3.1 Schedtune example 1 84
4.3.2 Schedtune example 2 85
4.3.3 Schedtune example 3 85
4.3.4 SCHED_R and SCHED_D guidelines 86
4.4 The nice and renice commands 87
4.4.1 Running a command with nice 87
4.4.2 Changing the nice value on a running thread 89
4.5 Summary 89
4.5.1 The schedtune command 90
4.5.2 The nice and renice commands 90
4.6 Quiz 91
4.6.1 Quiz answers 91
4.7 Exercise 91
4.8 The ps command 93
4.8.1 Use of ps command in CPU usage study 93
4.8.2 Use of ps command in memory usage study 95
4.9 The bindprocessor command 99
4.10 The emstat command 101
4.11 The tprof command 103
4.11.1 Using tprof general report 103
4.11.2 Using tprof on a program 105

Chapter 5. LVM and JFS performance tools 107
5.1 Overview 107
5.2 LVM performance analysis using lslv 108
5.2.1 Logical volume attributes 108
5.2.2 Logical volume fragmentation 112
5.2.3 Logical volume allocation 113
5.2.4 Highest LVM performance 115
5.3 LVM and file system monitoring 115
5.3.1 The filemon command 115
5.3.2 Report analysis 117
5.3.3 Typical AIX system behavior 123
5.4 File System Performance 124
5.4.1 AIX File System Organization 124
5.4.2 The fileplace command 126
5.4.3 File system de-fragmentation 128
5.5 General recommendations on I/O performance 128
5.6 Overhead of using performing tools 130
5.7 Commands summary 131
5.7.1 filemon 131

5.7.2 fileplace . . . 131
5.7.3 lslv . . . 132
5.8 Quiz . . . 133
5.8.1 Answers . . . 133
5.9 Exercises . . . 133

Chapter 6. Network performance tools . . . 135
6.1 Overview . . . 135
6.2 Adapter Transmit and Receive Queue Tuning . . . 137
6.3 Protocols tuning . . . 139
6.4 Network monitoring tools . . . 141
6.4.1 The vmstat command . . . 141
6.4.2 The ping command . . . 142
6.4.3 The netstat command . . . 142
6.4.4 The netpmon command . . . 145
6.4.5 The tcpdump and iptrace commands . . . 146
6.5 Network tuning tools . . . 149
6.6 Name resolution . . . 151
6.7 NFS tuning . . . 151
6.7.1 NFS server performance . . . 151
6.7.2 NFS client performance considerations . . . 153
6.7.3 Mount options . . . 155
6.8 Commands . . . 155
6.8.1 netstat . . . 155
6.8.2 tcpdump . . . 156
6.8.3 iptrace . . . 156
6.8.4 ipreport . . . 157
6.9 References . . . 157
6.10 Quiz . . . 158
6.10.1 Answers . . . 158
6.11 Exercises . . . 158
6.12 Collecting data using the sar command . . . 159
6.12.1 The sar command . . . 163
6.12.2 The sadc command . . . 170
6.12.3 The sa1 and sa2 commands . . . 170
6.13 Quiz . . . 171
6.13.1 Answers . . . 171
6.14 Exercises . . . 171
6.15 topas command . . . 173
6.16 The vmtune command . . . 177
6.17 Collecting data using the svmon command . . . 183
6.17.1 The svmon global report . . . 183
6.17.2 The svmon user report . . . 186

6.17.3 The svmon process report 189
6.17.4 The svmon segment report 191
6.17.5 The svmon detailed segment report 194
6.17.6 The svmon command report 196
6.17.7 The svmon workload management class report 199
6.17.8 The svmon command flags 200
6.18 Quiz 203
6.18.1 Answers 203
6.19 Exercises 203
6.20 Collecting data using the vmstat command. 205
6.21 Quiz 213
6.21.1 Answers 213
6.22 Exercises 213

Chapter 7. Performance commands: iostat, lockstat 215
7.1 The iostat command 215
7.1.1 Historical disk I/O 216
7.1.2 TTY and CPU utilization report 217
7.1.3 iostat on SMP systems 219
7.1.4 Disk utilization report 220
7.2 The lockstat command 222

Chapter 8. CPU testcase 227

Chapter 9. Examining I/O performance problem scenarios 231
9.1 I/O Bottlenecks. 231
9.1.1 Scenario1 231
9.1.2 Scenario2 232
9.2 Logical volume performance problem scenarios 234
9.2.1 Logical Volume Fragmentation scenario 234
9.2.2 Monitoring scenario using filemon 235
9.2.3 Logical volume allocation problem scenario 235

Chapter 10. I/O performance problem scenario two. 239
10.1 The system output 239
10.2 The output investigation 241
10.2.1 The vmstat command output investigation 241
10.2.2 The iostat command output investigation 241
10.3 Recommendations 242

Chapter 11. Paging performance problem scenario 243
11.1 The system output 243
11.2 The output investigation 249
11.2.1 The kthr (kernel thread) column 250

11.2.2 The memory column 250
11.2.3 The page column 250
11.2.4 The faults column 251
11.2.5 The cpu column 251
11.3 Recommendations 252

Appendix A. Error log 253
A.1 Overview 253
A.2 Managing Error Log 254
A.2.1 Configuring Error Logging. 254
A.2.2 Clearing Error log 255
A.3 Reading error logs in details 256
A.3.1 The errpt command output 257
A.3.2 Examples of formatted output from errpt command 258
A.4 Commands 260
A.4.1 errpt 260
A.5 References 261
A.6 Quiz. 262
A.6.1 Answers 262
A.7 Exercises 262

Appendix B. Using the additional material 263
B.1 Using the CD-ROM or diskette 263
B.1.1 System requirements for using the CD-ROM or diskette 263
B.1.2 How to use the CD-ROM or diskette 263
B.2 Locating the additional material on the Internet 263
B.3 Using the Web material. 264
B.3.1 System requirements for downloading the Web material 264
B.3.2 How to use the Web material 264

Appendix C. Special notices 265

Appendix D. Related publications. 269
D.1 IBM Redbooks 269
D.2 IBM Redbooks collections 269
D.3 Other resources 269
D.4 Referenced Web sites 270

How to get IBM Redbooks 271
IBM Redbooks fax order form 272

Abbreviations and acronyms 273

Index 275

IBM Redbooks review 285

Figures

1. smitty list_software output 40
2. smitty install_all 41
3. General performance tuning flowchart. 49
4. Process state 50
5. VMM segments 55
6. VMM from a process perspective 57
7. VMM memory registers 58
8. Logical Volume Device Driver 64
9. Dependencies in a volume group 65
10. Network parameters. 71
11. Performance tuning flowchart 74
12. Run queue on AIX version prior to 4.3.3 77
13. AIX Version 4, 128 run queues 78
14. Run queue on AIX Version 4.3.3 82
15. CPU penalty example 86
16. Disk, LVM and file system levels 107
17. LVM intra disk positons 111
18. JFS file system organization. 125
19. UDP/TCP/IP data flow 137
20. topas command output 173
21. SMIT screen for changing characteristics of operating system. 217
22. smitty errpt output 257

Tables

1. Performance tools overview 37
2. Performance toolbox releases 38
3. General installp summary 44
4. Commonly used flags of the lslpp command 45
5. Commonly used flags of the lppchk command 45
6. Hardware and logical resources 48
7. Processes and threads 51
8. VMM related output from the vmstat command 61
9. Some useful schedtune flags 86
10. Some nice flags 89
11. Some renice flags 89
12. Some useful schedtune flags 90
13. Some useful nice flags 90
14. Some renice flags 91
15. CPU related ps output 93
16. Memory related ps output 95
17. Commonly used flags of the filemon comman 131
18. Commonly used flags of the fileplace comman 132
19. Commonly used flags of the filemon comman 132
20. Some useful netstat flags 156
21. Some useful tcpdump commands 156
22. Some useful iptrace commands 157
23. Some useful ipreport flags 157
24. The sar command flags 164
25. The vmtune command flags 178
26. The svmon command flags 200
27. Commonly used flags of the iostat comman 215
28. Flags of the lockstat comman 223
29. Commonly used flags of the errpt command 261

Preface

The AIX and RS/6000 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 RS/6000. A complete set of professional certifications are available. They include:

- IBM Certified AIX User
- IBM Certified Specialist - AIX System Administration
- IBM Certified Specialist - AIX System Support
- IBM Certified Specialist - AIX HACMP
- IBM Certified Specialist - Business Intelligence for RS/6000
- IBM Certified Specialist - Domino for RS/6000
- IBM Certified Specialist - RS/6000 Solution Sales
- IBM Certified Specialist - RS/6000 SP and PSSP V3
- IBM Certified Specialist - RS/6000 SP
- RS/6000 SP - Sales Qualification
- IBM Certified Specialist - Web Server for RS/6000
- IBM Certified Advanced Technical Expert - RS/6000 AIX

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. These job incumbents bring a wealth of experience into the development process, thus, making the exams much more meaningful than the typical test that only captures classroom knowledge. These experienced subject matter experts ensure the exams are relevant to the *real world* and that 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 Redbook is designed as a study guide for professionals wishing to prepare for the Installation and System Recovery certification exam as a selected course of study in order to achieve: IBM Certified Advanced Technical Expert - RS/6000 AIX.

This 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 in the exam.

This publication does not replace practical experience or is 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 call IBM at 1-800-426-8322 or visit the 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.

Thomas C. Cederlöf is a Education Specialist at IBM Learning Services in Sweden. After working various professions, he was hired as a System Support Specialist in April 1997 at the Nordic AIX Competence Center. After earning his Advanced Technical Expert Certification in 1998 he worked with level 2 support in Scandinavia and the Baltic States, and participated also in the itrans program in 1999. Since January 2000 he is the main instructor for the AIX curriculum in Sweden.

André de Klerk is a Senior IT Specialist at IBM Global Services in South Africa. He has been working for IBM since May 1996. He started his career as a field technician in 1991 and has performed various support roles including application support and customer consulting. Currently he is team leader for the Midrange UNIX team at IGS SA.

Thomas Herlin is an Advisory IT Specialist at IBM Global Services in Denmark. He has been working for IBM since May 1998. Before joining IBM he has been working as a Software Engineer designing and developing programs on UNIX platforms. His areas of expertise include system architecture and system integration of AIX based solutions. He is also a certified SAP technical consultant.

Tomasz Ostaszewski is a computer network architect. He works for Prokom Software SA in Poland - IBM Business Partner. Prokom is the largest IT solution provider in Poland. They offer total solutions which include application development or third party vendor support. He has three years of experience in RS/6000 and AIX. Currently he is working on network project for an insurances company.

The project that produced this publication was managed by:

Scott Vetter IBM Austin

Thanks to:

Bill Hughes Program Manager, AIX & RS/6000 Certification

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

Shawn Mullen IBM Austin

Malin Cederberg ILS Sweden

Karl Borman ILS Austin

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks review” on page 285 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Certification overview

This chapter provides an overview of the skill requirements needed to obtain an IBM AIX Specialist certification. The following chapters are designed to provide a comprehensive review of specific topics that are essential for obtaining the certification.

1.1 IBM Certified Advanced Technical Expert - RS/6000 AIX

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 RS/6000 AIX.

To attain the IBM Certified Advanced Technical Expert - RS/6000 AIX certification, you must pass four tests.

One test is the prerequisite in either AIX System Administration or AIX System Support. The other three tests are selected from a variety of AIX and RS/6000 topics. These requirements are explained in greater detail in the sections that follow.

1.1.1 Required prerequisite

Prior to attaining the IBM Certified Advanced Technical Expert - RS/6000 AIX certification, you must be certified as either an:

- IBM Certified Specialist - AIX System Administration

  or

- IBM Certified Specialist - AIX System Support

1.1.2 Recommended prerequisite

A minimum of six to twelve months experience in performing in-depth analysis and applying complex AIX concepts in a variety of areas within RS/6000 AIX is a recommended prerequisite.

1.1.3 Registration for the certification exam

For information about *How to Register* for the certification exam, please visit the following Web site:

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

1.1.4 Core requirement (select three of the following tests)

You will receive a Certificate of Proficiency for tests when passed.

1.1.4.1 AIX V4.3 Installation and System Recovery

The following objectives were used as a basis when the certification test 183 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 1 - Installation and software maintenance

- Install or migrate the operating system
- Install a licensed program product
- Remove an OPP or an LPP from the system
- Update a system
- Apply a selective fix
- Identify and resolve network install problems

Section 2 - System backup and restore

- Perform a complete backup of the system
- Implement backup using relative and absolute path
- Create a mksysb
- Understand advanced mksysb concepts
- Restore files

Section 3 - System initialization (boot) failures

- Understand concepts of system initialization
- Diagnose the cause of a system initialization failure
- Resolve a system initialization failure

Section 4 - File systems and LVM recovery

- Perform problem determination on a filesystem
- Determine a suitable procedure for replacing a disk
- Resolve problems caused by incorrect actions taken to change a disk drive
- Create a new volume group
- Create a logical volume

- Understand LVM concepts
- Resolve a complex LVM problem

1.1.4.2 AIX V4.3 Performance and System Tuning

The following objectives were used as a basis when the certification test 184 was developed.

Preparation for this exam is the topic of *IBM Certification Study Guide - AIX Performance and System Tuning*, SG24-6184.

Section 1 - Performance Tools & Techniques

- Use the `iostat` command
- Use the `filemon` command
- Use the `tprof` command
- Use the `netpmon` command
- Interpret `iostat` output
- Interpret `lsps` output
- Interpret `netstat` output
- Interpret `vmstat` output
- Know about perfpmr
- Know about performance diagnostic tool
- Look at run queue
- Look at system calls

Section 2 - Correcting performance problems

- Correct disk bottlenecks
- Correct NFS bottlenecks
- Correct network bottlenecks
- Correct communications adapter bottlenecks
- Understand random write-behind concepts
- Understand async I/O performance concepts
- Understand VMM I/O pacing
- Understand file fragmentation
- Understand logical volume fragmentation

Section 3 - VMM

- Identify and correct VMM performance problems
- Correct paging problems
- Know about Tuning File Memory Usage
- Know about memory load control
- Understand Page Space Allocation issues

Section 4 - Multiprocessor and process scheduling

- Know SMP commands
- Use the `bindprocessor` command
- Enable, disable, and show status of processors
- List CPU utilization per processor
- Know about `ps` command and threads
- Understand locking issues in SMP
- Know about process scheduling
- Understand priority calculations
- Understand the effect of schedtune on priorities

Section 5 - Tuning and customization

- Tune a system for optimum performance
- Use the `no` command
- Customize a LV for optimum performance
- Configure system parameters
- Tune network parameters
- Determine when application tuning is needed
- Understand real-time tuning
- Understand disk striping
- Tune I/O performance with `vmtune`
- Understand RAID performance issues
- Perform capacity planning
- Understand memory usage

1.1.4.3 AIX V4.3 Problem Determination Tools and Techniques

The following objectives were used as a basis when the certification test 185 was developed.

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

Section 1 - System dumps

- Create a system dump
- Understand valid system dump devices
- Determine the location of system dump data
- Identify the status of a system dump by the LED codes
- Identify appropriate action to take after a system dump
- Determine if a system dump is successful
- Use the `snap` command

Section 2 - Crash

- Understand the use and purpose of the crash command
- Verify the state of a system dump
- Show the stack trace using crash
- Use the `stat` subcommand in crash
- Manipulate data in the process table
- Interpret crash stack trace output
- Interpret crash process output
- Interpret crash TTY output

Section 3 - Trace

- Start and stop trace
- Run trace
- Report trace information
- Interpret trace output
- Use trace to debug process problems

Section 4 - File system and performance PD tools

- Use tools to identify and correct corrupted file systems
- Understand file system characteristics

- Resolve file system mounting problems
- Repair corrupted file systems
- Use `vmstat` command
- Use `iostat` command
- Use `filemon` command

Section 5 - Network problem determination

- Use PD tools to identify network problems
- Resolve a network performance problem
- Correct problem with host name resolution
- Diagnose the cause of a problem with NFS mounts
- Diagnose the cause of a routing problem
- Resolve a router problem

Section 6 - Error logs and diagnostics

- Use error logging
- Interpret error reports
- Invoke and use diagnostic programs

Section 7 - Other problem determination tools

- Set breakpoints using `dbx`
- Step through a program using `dbx`
- Run a program with arguments using `dbx`
- Read core files and locate traceback
- Debug problem using core files
- Read shell scripts
- Debug shell script problems

1.1.4.4 AIX V4.3 Communications

The following objectives were used as a basis when the certification test 186 was developed.

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

Section 1 - TCP/IP implementation

- Know TCP/IP concepts

- Understand TCP/IP broadcast packets
- Use and implement name resolution
- Understand TCP/IP protocols
- Know IP address classes
- Use interfaces available in LAN communications
- Understand the relationship between an IP address and the network interface
- Log into remote hosts using telnet and rologin
- Construct /etc/hosts.equiv and ~/.rhosts for trusted users
- Transfer files between systems using ftp or tftp
- Run commands on remote machines

Section 2 - TCP/IP: DNS implementation

- Setup a primary name server
- Setup a secondary name server
- Setup a client in a domain network

Section 3 - Routing: implementation

- Apply knowledge of the IP routing algorithm
- Setup and use the routing table and routes
- Implement and use subnet masking

Section 4 - NFS: implementation

- Manipulate local and remote mounts using the automounter
- Understand NFS daemons and their roles
- Configure and tune an NFS server
- Configure and tune an NFS client
- Setup a file system for mounting
- Understand the /etc/exports file
- Invoke a predefined mount.

Section 5 - NIS: implementation

- Understand the various NIS daemons
- Implement NIS escapes
- Create NIS map files

- Transfer NIS maps

Section 6 - Network problem determination

- Diagnose and resolve TCP/IP problems
- Diagnose and resolve NFS problems
- Diagnose and resolve NIS problems

Section 7 - Hardware related PD (modems)

- Determine appropriate diagnostic approach to resolve a modem connection problem
- Resolve communication configuration problems

1.1.4.5 HACMP for AIX V4.2

The following objectives were used as a basis when the certification test 167 was developed.

Preparation for this exam is the topic of *IBM Certification Study Guide - AIX HACMP*, SG24-5131.

Section 1 - Pre-installation

- Conduct a planning session
    - Set customer expectations at the beginning of the planning session
    - Gather customer's availability requirements
    - Articulate tradeoffs of different HA configurations
    - Assist customer in identifying HA applications
- Evaluate customer environment and tailorable components
    - Evaluate configuration and identify Single Points of Failure (SPOF)
    - Define and analyze NFS requirements
    - Identify components affecting HACMP
    - Identify HACMP event logic customizations
- Plan for installation
    - Develop disk management modification plan
    - Understand issues regarding single adapter solutions
    - Produce a test plan

Section 2 - HACMP implementation

- Configure HACMP solutions

  - Install HACMP code
  - Configure IP Address Takeover (IPAT)
  - Configure non IP heartbeat paths
  - Configure network adapter
  - Customize/tailor AIX
  - Set up shared disk (SSA)
  - Set up shared disk (SCSI)
  - Verify a cluster configuration
  - Create an application server
- Setup event notification
  - Set up event notification and pre/post event scripts
  - Setup error notification
- Post configuration activities
  - Configure client notification and ARP update
  - Implement test plan
  - Create a snapshot
  - Create a customization document
- Testing and Troubleshooting
  - Troubleshoot failed IPAT failover
  - Troubleshoot failed shared volume groups
  - Troubleshoot failed shared volume groups
  - Troubleshoot failed network configuration
  - Troubleshoot failed shared disk tests
  - Troubleshoot failed application
  - Troubleshoot failed pre/post event scripts
  - Troubleshoot failed error notifications
  - Troubleshoot errors reported by cluster verification

Section 3 - System management

- Communicate with customer
  - Conduct turnover session
  - Provide hands-on customer education

- Set customer expectations of their HACMP solution's capabilities
- Perform systems maintenance
  - Perform HACMP maintenance tasks (PTFs, adding products, replacing disks, adapters)
  - Perform AIX maintenance tasks
  - Dynamically update cluster configuration
  - Perform testing and troubleshooting as a result of changes

1.1.4.6 RS/6000 SP and PSSP V2.4

The following objectives were used as a basis when the certification test 178 was developed.

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

Section 1 - Implementation and planning

- Validate software/hardware capability and configuration.
  - Determine required software levels (for example., version, release, and modification level).
  - Determine the size, model and location of the control workstation.
  - Define disk, memory, and I/O including disk placement.
  - Determine disk space requirements.
  - Understand multi-frame requirements and switch partitioning.
  - Determine the number and type of nodes needed (including features).
  - Determine the number of types of I/O devices (for example, SCSI, RAID, SSA, etc.) needed.
  - Configure external I/O connections.
  - Determine additional network connections required.
  - Create the logical plan for connecting into networks outside the SP.
  - Identify the purpose and bandwidth of connections.
- Plan implementation of key aspects of TCP/IP networking in the SP environment.
  - Create specific host names (both fully qualified and aliases), TCP/IP address,
  - Netmask value and default routes.

    - Determine the mechanism (for example, /etc/hosts, NIS, DNS) by which they will be made available across the system.
    - Choose the IP name/address resolver.
- Determine the appropriate common, distributed, and local files/file systems.
    - Determine the physical locations of the file system and home directories.
    - Determine the number of types of I/O devices (for example, SCSI, RAID, SSA, etc.) needed.
    - Configure internal I/O.
    - Determine the mechanism (for example, NFS, AFS, DRS, local) by which they will be made available across the system.
- Configure and administer the Kerberos Authentication subsystem and manage user IDs on the SP system.
    - Define administrative functions.
    - Determine the Kerberos administration ID.
    - Define Administrative functions
    - Understand the options of end-user management.
    - Understand how to administer authenticated users and instances.
- Define a backup/recovery strategy for the SP which supports node images, control workstation images, applications, and data.
    - Determine backup strategy and understand the implications of multiple unique mksysb images.

Section 2 - Installation and configuration

- Configure an RS/6000 as an SP control workstation.
    - Verify the control workstation system configuration.
    - Configure TCP/IP network on the control workstation.
    - Install PSSP.
    - Load the SDR with SP configuration information.
    - Configure the SP System Data Repository.
    - Verify control workstation software.
    - Configure TCP/IP name resolution (for example, /etc/passwd, DNS, NIS).

- Perform network installation of images on nodes, using any combination of boot/install servers.
  - Install the images on the nodes.
  - Create boot/install servers
- Exercise the SP system resources to verify the correct operation of all required subsystems.
  - Verify all network connections.
  - Verify internal and external I/O connections.
  - Verify switch operations

Section 3 - Application enablement

- Determine whether LoadLeveler would be beneficial to a given SP system configuration.
  - Understand the function of LoadLeveler.
- Define and implement application specific FS, VG, and VSDs for a parallel application.
  - Define application-specific file systems, logical volumes, volume groups, or VSDs.
  - Implement application-specific file systems, logical volumes, volume groups, or VSDs.
- Install and configure problem management tools (for example, event manager, problem manager, perspectives)
  - Install and Configure user-management tools.

Section 4 - Support

- Utilize Problem Determination methodologies (for example, HOSTRESPONDS, SWITCHRESPONDS, error report, log files, DAEMONS, GUIs).
  - Handle resolution of critical problems.
  - Conduct SP-specific problem diagnosis.
  - Interpret error logs that are unique to SP.
- Isolate cause of degraded SP performance, and tune the system accordingly.
  - Understand performance analysis and tuning requirements

1.1.4.7 RS/6000 SP and PSSP V3

The following objectives were used as a basis when the certification test 188 was developed.

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

Section 1 - Implementation planning

- Validate software/hardware capability and configuration.
  - Determine required software levels (for example, version, release, and modification level)
  - Determine the size, model and location of the control workstation.
  - Define disk, memory, and I/O including disk replacement.
  - Define disk space requirements.
  - Understand multi-frame requirements and switch partitioning.
  - Determine the number and types of nodes needed (including features).
  - Determine the number and types of I/O devices (for example, SCSI, RAID,SSA, etc.) needed (including features).
  - Configure external I/O connections.
  - Determine additional network connections required.
  - Create the logical plan for connecting into networks outside the SP.
  - Identify the purpose and bandwidth of connections.
  - Determine if boot/install servers are needed and, if needed, where they are located.
- Implement key aspects of TCP/IP networking in the SP environment.
  - Create specific host names (both fully qualified and aliases), TCP/IP address, Netmask value and default routes.
  - Determine the mechanism (for example, /etc/hosts, NIS, DNS) by which they will be made available across the system.
  - Determine SP Ethernet topology (segmentation, routing).
  - Determine TCP/IP addressing for switch network.
- Determine the appropriate common, distributed and/or local files/file systems.
  - Determine the physical locations of the file system and home directories.

    - Determine the mechanism (for example, NFS, AFS, DRS, local) by which they will be made available across the system.
- Define a backup/recovery strategy for the SP which supports node image(s), control workstation images, applications, and data.
    - Determine backup strategy including node and CWS images.
    - Determine backup strategy and tools for application data.

Section 2 - Installation and configuration

- Configure an RS/6000 as an SP control workstation.
    - Verify the control workstation system configuration.
    - Configure TCP/IP network on the control workstation.
    - Install PSSP.
    - Configure the SDR with SP configuration information.
    - Verify control workstation software.
- Perform network installation of images on nodes, using any combination of boot/install servers.
    - Install the images on the nodes.
    - Define and configure boot/install servers.
    - Check SDR information.
    - Check RSCT daemons (hats, hags, haem).
- Thoroughly exercise the SP system resources to verify correct information of all required subsystems.
    - Verify all network connections.
    - Verify switch operations.
- Configure and administer the Kerberos Authentication subsystem and manage user IDs.
    - Plan and configure Kerberos functions and procedures.
    - Configure the Kerberos administration ID.
    - Understand and use the options of end-user management.
- Define and configure system partition and perform switch installation.

Section 3 - Application Enablement

- Determine whether additional SP-related products (for example, Loadleveler, PTPE, HACWS, NetTAPE, CLIOS) would be beneficial.
- Understand the function of additional SP-related products.

- Define and implement application-specific file systems, logical volumes, VGs and VSDs.
- Install and configure problem management tools (for example, event manager, problem manager, perspectives).
  - Define and manage monitors.

Section 4 - Ongoing support

- Perform software maintenance.
  - Perform system software recovery.
  - Upgrade and migrate system software (applying PTFs, migration).
- Perform SP reconfiguration.
  - Add frames.
  - Add nodes.
  - Migrate nodes.
  - Add/replace switch.
- Utilize Problem Determination methodologies (for example, HOSTRESPONDS,SWITCHRESPONDS, error report, log files, DAEMONS,GUIS).
  - Interpret error logs that are unique to the SP.
  - Diagnose networking problems.
  - Diagnose host response problems.
  - Diagnose switch-specific problems.
- Isolate cause of degraded SP performance and tune the system accordingly.
  - Understand Performance analysis and tuning requirements.

1.2 Certification education courses

Courses are offered to help you prepare for the certification tests. These courses are recommended, but not required, before taking a certification test. At the publication of this guide, the following courses are available. For a current list, please visit the following Web site:

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

AIX Version 4 System Administration	
Course Number	Q1114 (USA), AU14 (Worldwide)
Course Duration	Five days
Course Abstract	Learn the basic system administration skills to support AIX RS/6000 running the AIX Version 4 operating system. Build your skills in configuring and monitoring a single CPU environment. Administrators who manage systems in a networked environment should attend additional LAN courses.
Course Content	•Install the AIX Version 4 operating system, software bundles, and filesets •Perform a system startup and shutdown •Understand and use AIX system management tools •Configure ASCII terminals and printer devices •Manage physical and logical volumes •Perform file systems management •Create and manage user and group accounts •Use backup and restore commands •Use administrative subsystems, including cron, to schedule system tasks and security to implement customized access of files and directories

AIX Version 4.3 Advanced System Administration	
Course Number	Q1116 (USA), AU16 (Worldwide)
Course Duration	Five days
Course Abstract	Learn how to identify possible sources of problems on stand-alone configurations of the RS/6000 and perform advanced system administration tasks.
Course Content	•Identify the different RS/6000 models and architects •Explain the ODM purpose for device configuration •Interpret system initialization and problems during the boot process •Customize authentication and set up ACLs •Identify the TCB components, commands, and their use •Obtain a system dump and define saved data •Identify the error logging facility components and reports •List ways to invoke diagnostic programs •Customize a logical volume for optimal performance and availability •Manage a disk and the data under any circumstance •Use the standard AIX commands to identify potential I/O, disk, CPU, or other bottlenecks on the system •Customize SMIT menus and define how SMIT interacts with the ODM •Define the virtual printer database and potential problems •List the terminal attributes and create new terminfo entries •Define the NIM installation procedure

AIX Version 4 Configuring TCP/IP and Accessing the Internet	
Course Number	Q1107 (USA), AU07 or AU05 (Worldwide)
Course Duration	Five days
Course Abstract	•Learn how to perform TCP/IP network configuration and administration on AIX Version 4 RS/6000 systems. •Learn the skills necessary to begin implementing and using NFS, NIS, DNS, network printing, static and dynamic routing, SLIP and SLIPLOGIN, Xstations, and the Internet.
Course Contents	•Describe the basic concepts of TCP/IP protocols and addressing •Explain TCP/IP broadcasting and multicasting •Configure, implement, and support TCP/IP on an IBM RS/6000 system •Use networking commands for remote logon, remote execution, and file transfer •Configure SLIP and SLIPLOGIN •Use SMIT to configure network printing •Connect multiple TCP/IP networks using static and dynamic routing •Implement DNS, NFS, and NIS •Perform basic troubleshooting of network problems •Configure an Xstation in the AIX environment •Explain how to access Internet services •Understand and support TCP/IP •Plan implementation of NFS •Support LAN-attached printers •Support AIX networking •Determine network problems •Implement network file systems

1.3 Education on CD: IBM AIX Essentials

The new IBM AIX Essentials series offers a dynamic training experience for those who need convenient and cost-effective AIX education. The series consists of five new, content rich, computer-based multimedia training courses based on highly acclaimed, instructor-led AIX classes that have been successfully taught by IBM Education and Training for years.

To order, and for more information and answers to your questions:

- In the U.S., call 800-IBM-TEACH (426-8322) or use the online form at the following URL: `http://www-3.ibm.com/services/learning/aix/#order`
- Outside the U.S., contact your IBM Sales Representative or
- Contact an IBM Business Partner.

Chapter 2. 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 well 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 sufficient resources can be acquired well before they are required.

AIX provides several tools which cover these areas. Not all of these tools come with the AIX Base Operating System. Some of them are part of the Performance Toolbox software as shown in Table 1 on page 37.

Table 1. Performance tools overview

Software product	Tool
Base Operating System	vmstat, iostat, sar, ps, netstat, nfsstat, no, nfso, trace, prof, gprof, perfpmr, schedtune, vmtune
perfagent.tools fileset (not part of BOS but part of Server bundle)	bf, bfrpt, fdpr,filemon, fileplace, genkex, genkld, genld, gennames, ipfilter, lockstat, netpmon,pprof, rmss, stem, stripnm, svmon, syscalls, topas, tprof
Performance Toolbox Agent	xmservd, filtd
Performance Toolbox Manager	xmperf, 3dmon, 3dplay, azizo, exmon, chmon

The Performance Toolbox is a Motif-based, AIX Licensed Program Product (LPP) that consolidates AIX performance tools into 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 previous Performance Toolbox contained two filesets perfagent.server and perfagent.tool, causing installation difficulty. To improve this process, those pieces that are required to be built with the AIX kernel are moved into the perfagent.tools fileset. Then the agent becomes mainly an interface routine to those pieces.

The perfagent.tools fileset is shipped with the AIX Version 4.3.2 base. For AIX Version 4.3.2, the Performance Toolbox Agent will prerequisite perfagent.tools. So the .tools fileset must be installed first.

Table 2 on page 38 lists the various minimum fileset levels required with a particular AIX level.

Table 2. 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 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 version	Performance Agent	Performance Manager
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

As you can see the Performance Manager releases remain the same throughout AIX Version 4.1.5, 4.2.1, 4.3.1, 4.3.2. 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.

To get Performance Toolbox Manager or Performance Toolbox Agent you should visit the following Web page `http://www.rs6000.ibm.com/`.

Before you install Performance Toolbox software you have to determine AIX level and maintenance level of your system. To determine the AIX maintenance level of the system, enter:

```
# oslevel
4.3.3.0
```

The current maintenance level of the system is 4.3.3.0. To list the names of known maintenance levels use:

```
# oslevel -q
Known Maintenance Levels
------------------------
4.3.3.0
4.3.2.0
4.3.1.0
```

To determine the filesets at levels later than the current AIX maintenance level, enter:

```
# oslevel -g
Fileset                                Actual Level       Maintenance Level
---------------------------------------------------------------------------
bos.docsearch.client.com                 4.3.3.0            4.3.2.0
perfagent.tools                          2.2.34.0           2.2.33.0
```

Now you know that the system is at 4.3.3.0 level. There are two filesets at the higher level and one of then is `perfagent.tools` fileset. Next, lets check if any of the Performance Toolbox packages are installed on your system:

```
# lslpp -l perf*
```

```
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  perfagent.html.en_US.data  2.2.1.0  COMMITTED  Performance User and .
Reference Guides - U.S. English
  perfagent.html.en_US.usergd
                            4.3.3.0  COMMITTED  Performance Toolbox Guides -
                                                 U. S. English
  perfagent.tools           2.2.34.0  COMMITTED  Local Performance Analysis&
                                                 Control Commands
```

You can also do this using `smitty list_software`:

```
                              COMMAND STATUS

Command: OK            stdout: yes          stderr: no

Before command completion, additional instructions may appear below.

[TOP]
  Fileset                       Level  State  Description
  ----------------------------------------------------------------------------
  perfagent.html.en_US.data  2.2.1.0    C     Performance User and Reference
                                              Guides - U.S. English
  perfagent.html.en_US.usergd
                             4.3.3.0    C     Performance Toolbox Guides - U.
                                              S. English
  perfagent.tools           2.2.34.0    C     Local Performance Analysis &
                                              Control Commands


State Codes:
[MORE...6]

F1=Help             F2=Refresh          F3=Cancel           F6=Command
F8=Image            F9=Shell            F10=Exit            /=Find
n=Find Next
```

Figure 1. smitty list_software output

As show there is only `perfagent.tools` fileset installed. If you have media with Performance Toolbox software you can check what it contains. You can use either `installp` command or as shown in the Figure 1 `smitty list_software`.command to list software on the installation media. The output from `installp` command apears like:

```
# installp -ld . -I
  Fileset Name                Level                      I/U Q Content
  ====================================================================
  perfagent.html.en_US.usergd 4.3.0.0                     I  N usr
#   Performance Toolbox Guides - U. S. English
```

```
  perfagent.server            2.2.30.0                        I  N usr,root
#   Performance Agent Daemons & Utilities

  perfmgr.common              2.2.1.0                         I  N usr
#   Performance Toolbox Manager - Common Support

  perfmgr.local               2.2.1.0                         I  N usr
#   Performance Toolbox Manager - Local Support

  perfmgr.network             2.2.0.0                         I  N usr
#   Performance Toolbox Manager - Network Support
```

The directory contains both components: Performance Toolbox Manager and Performance Toolbox Agent. Now you know what you need to be installed and you also know what you get on installation media. This is the time to install Performance Toolbox. The easiest way to install software is the `smitty install_all` command:

```
                  Install and Update from ALL Available Software

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
* INPUT device / directory for software                  /tmp
* SOFTWARE to install                                   []                      +
  PREVIEW only? (install operation will NOT occur)       no                     +
  COMMIT software updates?                               no                     +
  SAVE replaced files?                                   no                     +
  AUTOMATICALLY install requisite software?              yes                    +
  EXTEND file systems if space needed?                   yes                    +
  OVERWRITE same or newer versions?                      no                     +
  VERIFY install and check file sizes?                   no                     +
  DETAILED output?                                       no                     +
  Process multiple volumes?                              yes                    +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 2. smitty install_all

Do not commit software until you are sure that installation process does not impact the system. After installation check $HOME/smit.log for errors and run `lppchk` command to verify installation process. If every thing is OK you can commit your installation running `installp -c all`. If not, you should clean up a failed installation with `installp -C` command.

To check that you have the filesets installed for manager and agent part perform the following steps:

For agent part:

```
# 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 manager part:

```
# 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:

```
# 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
```

2.1 Commands

For a complete reference of the following command use the *AIX Version 4.3 Command Reference* or the online man pages.

2.1.1 installp

The `intsallp` command is a very useful and powerful 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 deinstall (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 Media:

```
installp { -l | -L } [ -eLogFile ] [ -d Device ] [ -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 3 on page 44 is a general summary of some useful `installp` flags.

Table 3. General installp summary

Flag	Description
-ac	Commit
-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	Input device
-l	List of installable filesets
-c	Commit an applied fileset
-C	Clean up after an failed installation
-u	Uninstall
-r	Reject an applied fileset
-p	Preview of installation
-e	Define an installation log
-F	Forced overwrite of same or newer version

2.1.2 lslpp

Display information about installed filesets or fileset updates. The command has the following syntax:

```
lslpp { -f | -h | -i | -L } ] [ -a ] [ FilesetName ... | FixID ... | all ]
```

Table 4. Commonly used flags of the lslpp command

Flag	Description
-a	Displays all the information about filesets specified when combined with other flags. Displays all the information about filesets specified when combined with other flags.
-f	Displays all the information about filesets specified when combined with other flags.
-h	Displays the installation and update history information for the specified fileset
-i	Displays the product information for the specified fileset.
-L	Displays the name, most recent level, state, and description of the specified fileset. Part information (usr, root, and share) is consolidated into the same listing.
-w	Lists fileset that owns this file

2.1.3 lppchk

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 5. 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.

2.2 References

The following publications contain more information about.

- *Performance Toolbox Version 1.2 and 2 for AIX: Guide and Reference.*
- *Installation Guide,* SC23-4112.
- *AIX Version 4.3 Problem Solving Guide and Reference*, SC23-4123.
- *AIX Version 4.3 Commands Reference, Volume 3*, SC23-4117.

2.3 Quiz

2.3.1 Answers

2.4 Exercises

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.

Chapter 3. Performance tuning - getting started

In this chapter will the following topics be covered:

- Introduction to concepts and tools
- Performance tuning flowchart

Generally speaking 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, in that the user response time and system interactions are concerns addressed during the design phase, not the overall system performance. This part of performance, optimization of code, is outside the scope of this publication. This publication will consider the system management aspects.

3.1 Introduction to concepts

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

Performance tuning

Performance tuning is the tuning of application and allocation of resources to best meet defined requirements and goals

From this definition of performance tuning, the tasks included can be listed in the following order:

1. Identify 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 objectives are usually either maximize response throughput within a given response time or maximize response time under a consistent workload.

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

2. Defining and prioritizing goals

   When discussing performance goals, a ‘fast system’ is not a appropriate objective. Depending on the task of the system, the goals will differ. Usually the goal is one of the following or a compromise of both - reduce response time or maximize system throughput. There will be trade offs. Sometimes the trade off is performance versus cost, sometimes the trade off is to increase the performance of some hardware or software at the expense of some other piece of hardware or software.

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 ares - physical resources and logical resources. Here are some examples of hardware with their logical resources:

Table 6. Hardware and logical resources

Hardware resource	Logical resource
CPU	Process time slice
Memory	Page frame
	Stacks
	Buffers
	Queues
	Tables
Disk Space	Logical Volumes
	File systems
	Files
Communication Lines	Packets
	Channels

4. Minimize resource requirements

   This can be accomplished 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 were the actual hands-on tuning will occur. Point one through three are more or less planning and researching.

5. Controlling allocation of resources

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

In the following section the classical performance tuning flowchart will briefly be discussed.

3.2 CPU bound

When investigating a performance problem, CPU constraint is probably the easiest to find. That is why most performance analysts start with checking for CPU constrains, and work their way down the flowchart shown in Table 3.

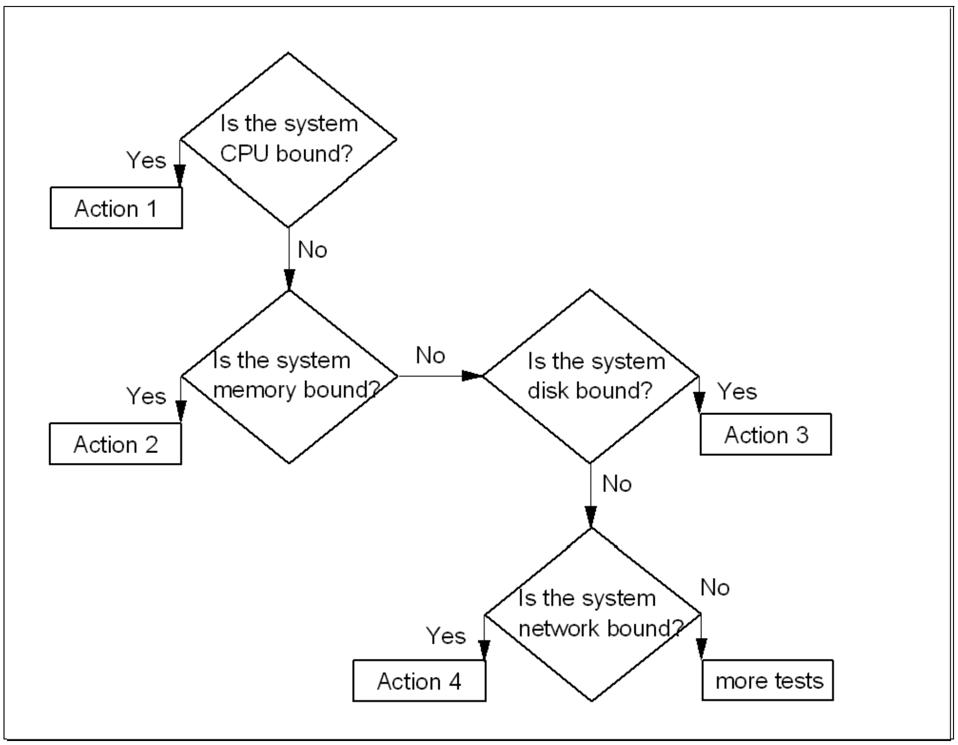

Figure 3. General performance tuning flowchart

If a system is CPU bound the cause should be searched for in the two entities using the CPU - processes and threads. The CPU handles only threads, and therefore a process has to have at least one thread. Usually (AIX Version 4), a process is multi-threaded, which means that a process can use multiple threads to accomplish its task. In Figure 4 the relationship between processes and threads is symbolized.

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

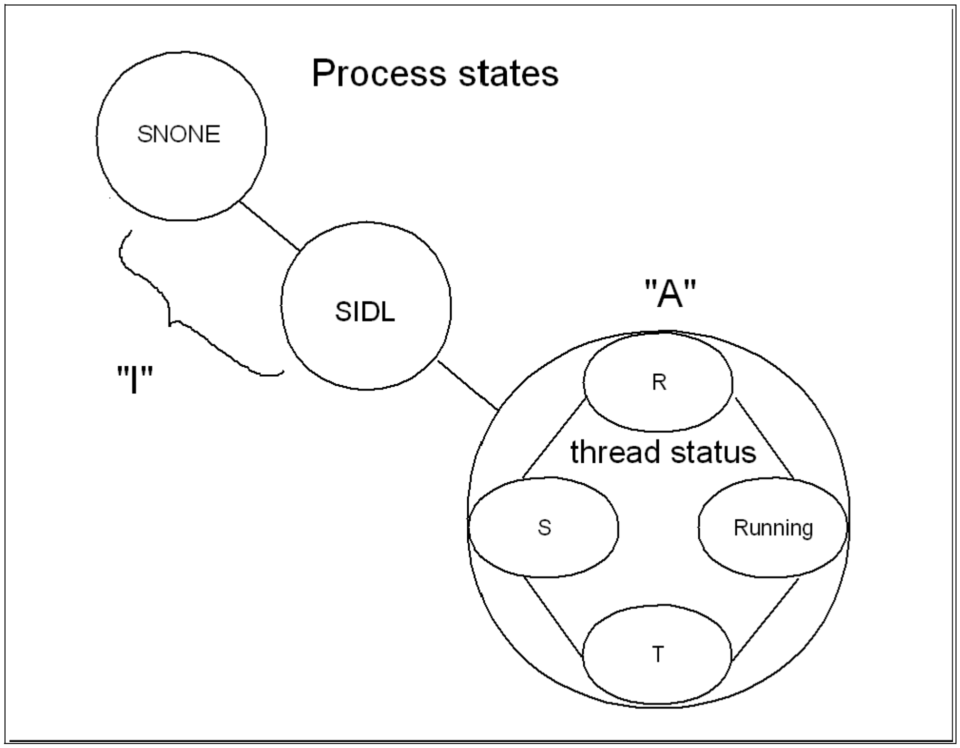

Figure 4. Process state

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 if it is waiting for 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, the priority calculation are all ways to tune (and misstune, if not carefully done) the CPU. More on the run queue and how the decide which thread is to be prioritized is discussed in Chapter 4, “CPU performance tools” on page 75.

When working with CPU tuning you have to know what can be tuned on a process level and what can be tuned on a thread level and chose accordingly. In Table 7 is a list of some process related properties and thread related properties.

Table 7. Processes and threads

Process properties:	**Thread properties:**
PID & PGID	TID
UID & GID	Stack
Environment	scheduling policy
Cwd	Pending signals
File descriptors	Blocked signals

For example a simple choice of shell for a process running in the background will change the process performance, because the ksh shell will add 4 to the nice value for background processes while the c shell does not do this.

When working with CPU performance tuning you have good use of historical performance information for comparison reasons. Usually performance is a subject to very personal and subjective view points. To avoid confusion, hard copies of performance statistics from a time when users did not complain, should be filed. A very useful tool for this task is the `sar` command.

sar

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 scripts collects data and saves the data for one week. Another useful feature with `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 on the `sar` command in “Collecting data using the sar command” on page 159 and in 6.12.1, “The sar command” on page 163.

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

time

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. 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 can be as 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 its 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 on the usage of the time command is found in “CPU testcase” on page 227.

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

vmstat

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
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 16998 14612   0   0   0   0    0   0 101   10   8 55  0 44  0
 0  1 16998 14611   0   0   0   0    0   0 411 2199  54  0  0 99  0
 0  1 16784 14850   0   0   0   0    0   0 412  120  51  0  0 99  0
 0  1 16784 14850   0   0   0   0    0   0 412   88  50  0  0 99  0
```

The `r` column shown threads in the R state, while the `b` column shown threads in S state, as shown in Figure 4 on page 50. The four right-hand columns are a breakdown in percentage of CPU time used on user threads, system threads, CPU idle time (running the wait process), and CPU idle time during which the system had outstanding disk/NFS I/O request. For more on the `vmstat` command, se “Collecting data using the vmstat command” on page 205.

If the system has bad performance because of a lot of threads on the run queue or threads waiting for I/O, then `ps` output will be useful in determine which process has used most CPU resources.

ps

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 “The ps command” on page 93, is the ps command output relevant from a CPU tuning perspective discussed.

When finding a run-away process the next step in the analysis is to find out what exactly in the process uses CPU. For this is a profiler needed. The AIX profiler of preference is `tprof`.

tprof

The `tprof` can be used for application tuning and for overall CPU utilization information gathering. The `tprof` command can be runned 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 ticks were spent in respective address space. If the user column shows high values, application tuning might be necessary. More on the `tprof` command in “The tprof command” on page 103.

When finding a process that cannot be optimized, another way to approach the problem is to lessens its priority in the run queue. This can be accomplished by grouping processes together into groups to be handled by the Workload Manager or by us of the `nice` and `renice` commands.

nice and renice

The `nice` command lets you run a command at a priority lower than the command's normal priority. You must have root user authority to run a command 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 tick, the nice value is stable and manipulated with the `nice` or `renice` command, as referred to in this book. The nice value can range from 0 to 39, with 39 being the lowest priority. For example, if a command normally runs with a default nice value of 20, specifying an increment of 5 runs the command at a lower priority, 25, and the command may run slower. More on the priorities and nice values in 4.1.1, “Priority calculation on AIX versions prior to 4.3.2” on page 76, 4.1.2, “Priority calculation on AIX version 4.3.2 and later” on page 79, and “The nice and renice commands” on page 87.

Finally there is the `schedtune` command. This command is mentioned last for a reason, do not manipulate the scheduler without thurough knowledge of the scheduler mechanism.

schedtune

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 on the scheduler and on the `schedtune` command is covered in Chapter 4.1, “The AIX scheduler” on page 75 and in “The schedtune command” on page 84.

3.3 Memory bound

Memory in AIX is handled by the Virtual Memory Manager (VMM). Virtual memory manager is a method by which real memory appears larger than its true size. The virtual memory system is composed of real memory plus physical disk space where portions of a file 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 5 on page 55:

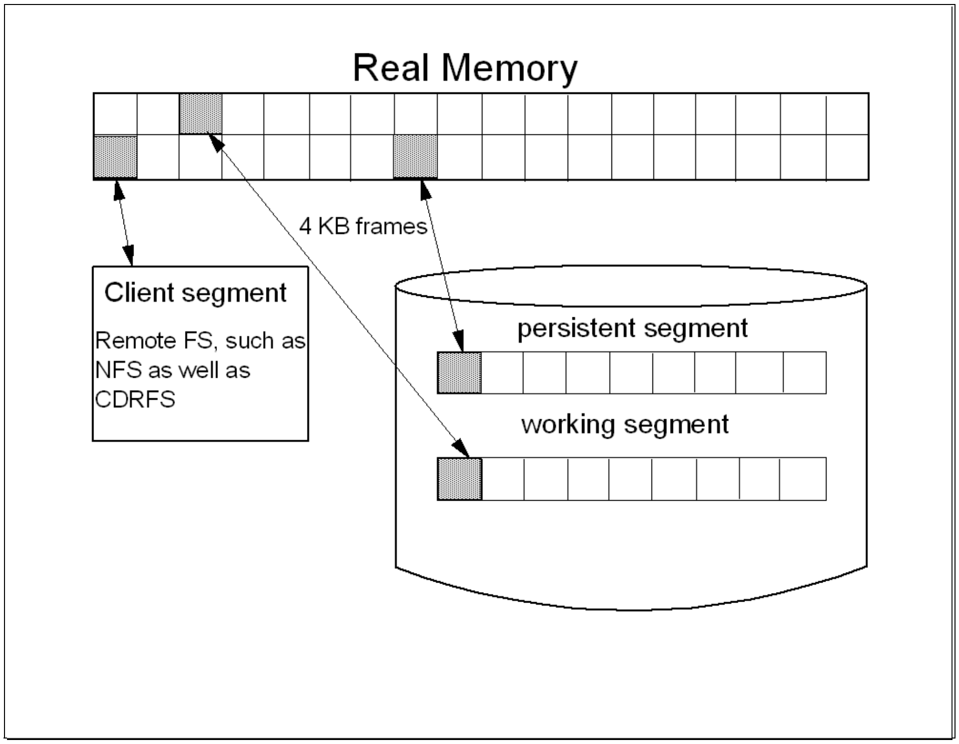

Figure 5. VMM segments

The three types of segments are as follows:

- *Persistent segment*

  Persists after use by a process and has (and use) 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 and/or file access is 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*

This is transitory and exists only during use by their process. They have no permanent disk storage location and are therefore stored to paging space if free page frames in real memory is needed. For example, kernel text segments and process stack are mapped to working segments.

- *Client segment*

  Segments of which 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 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* segment

  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 process, for example, processes can share code segments, yet have private data segments.

  The relationship is exemplified in Figure 6 on page 57.

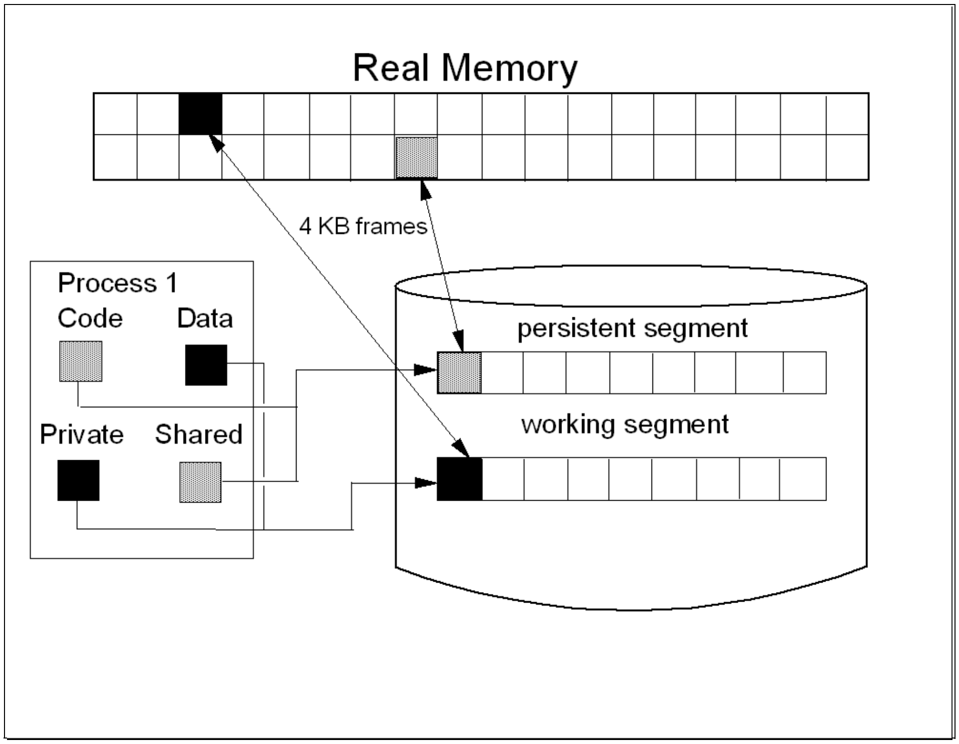

Figure 6. VMM from a process perspective

From a process point of view the memory is further divided into 16 *segment registers*. These segment registers are actually hardware registers located on the processor. When a process is active, the register contain the addresses of the 16 segments addressable by that process. Each segment contains a specific type of information, as shown in Figure 6 on page 57.

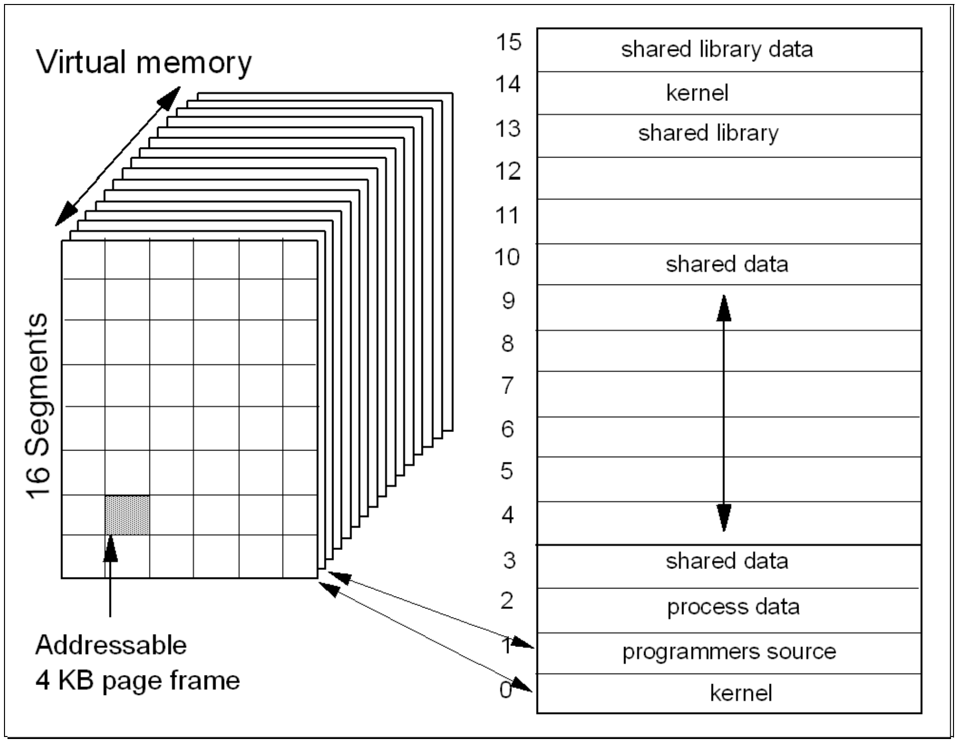

Figure 7. VMM memory registers

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.

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 is the data and instruction caches searched. Next is the *Translation Lookaside Buffer* (TLB) 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. As the table is large, there are indexes to this index. The *Hash Anchor Table* (HAT) links pages of related segments, to get a faster entry point to the main PTF.

From the page stealer perspective the memory is divided into *Computational memory* and *File memory*. The page stealer tries to balance these two types of memory usage when stealing pages. The page replacement algorithm can be manipulated.

- Computational memory are 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.

When starting a process, a slot has to be 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, continous pagein and pageout occurs. This condition is called trashing.

When discussing memory the allocation algorithm has to be mentioned. Here follows a discussion from *System Management Concepts: Operating System and Devices - Second edition*, 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 Version 4.3.2 and later, the late allocation algorithm is modified to further delay the allocation of paging space. As mentioned above, before 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 overcommitment 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 overcommitment. 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 Version 4.3.2 is also called 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 paging space because some of it may be back in RAM as well. If the page that was paged back in is 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` commands on Version 4.3.3. For more information on the differences between Version 4.3.2 and Version 4.3.3 refer to *Commands Reference - Volume 5*, SBOF-1877 and *System Management Concepts: Operating System and Devices*, SC23-4311.

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

vmstat

The `vmstat` command summarizes the total active virtual memory used by all of the processes in 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 have actually 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 might be short on memory or if some memory tuning needs to be done, run 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 might be a memory bottleneck. Having occasional non-zero values is not be a concern because paging is the main principle of virtual memory

From a VMM tuning perspective the middle (highlighted) columns are most interesting. They provide information about the usage 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 columns are explained as follows:

Table 8. 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 on the `vmstat` command, see “Collecting data using the vmstat command” on page 205.

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.

ps

The `ps` command can also be used to monitor 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 “The ps command” on page 93.

In the previous discussion, the paging space part in VMM was mentioned. The `lsps` command is an useful tool for paging space utilization checks.

lsps

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.

```
# 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 will give more detailed information on what process are using what segments of memory.

svmon

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 Version 4.3.2 and Version 4.3.3. This is discussed in more detail in “Collecting data using the svmon command” on page 183 and in *Performance Management Guide*.

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

vmtune

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 reboot of the system. More on the `vmtune` command is found in 6.16, “The vmtune command” on page 177 and in *Performance Management Guide*.

Sometimes testing of how much (or should we say little) memory is needed for a certain server load is done with the `rmss` command.

rmss

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 either of two 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.

More on the `rmss` command can be found in *Commands Reference - Volume 4*, SBOF-1877.

3.4 Disk Bound

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 Logical Volume Manager (LVM) controls disk resources by mapping data between a more simple and flexible logical views of storage space and the actual physical disks. The LVM does this using a layer of device driver code that runs above traditional disk device drivers.

The Logical Volume Manager (LVM) consists of the logical volume device driver (LVDD) and the LVM subroutine interface library. The logical volume device driver (LVDD) is a pseudo-device driver that manages and processes all I/O. 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 8 on page 64.

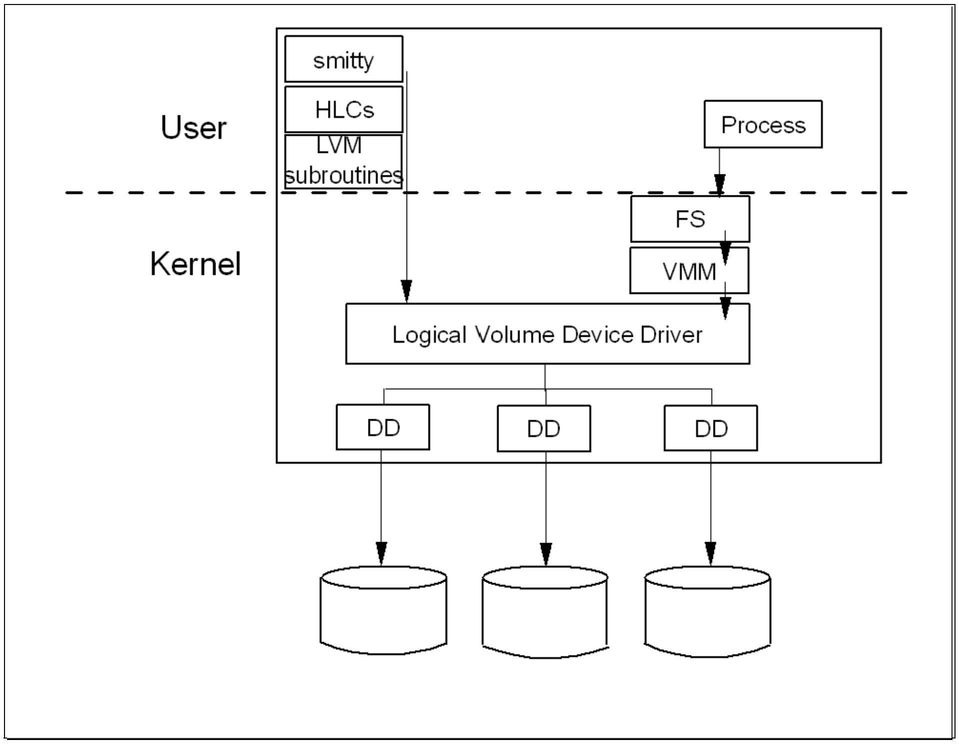

Figure 8. Logical Volume Device Driver

Each individual disk drive, called a physical volume (PV), has a name, 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 bigger 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.

The following illustration shows the relationship and dependencies between the logical picture of the Volume Group with its corresponding Physical layout.

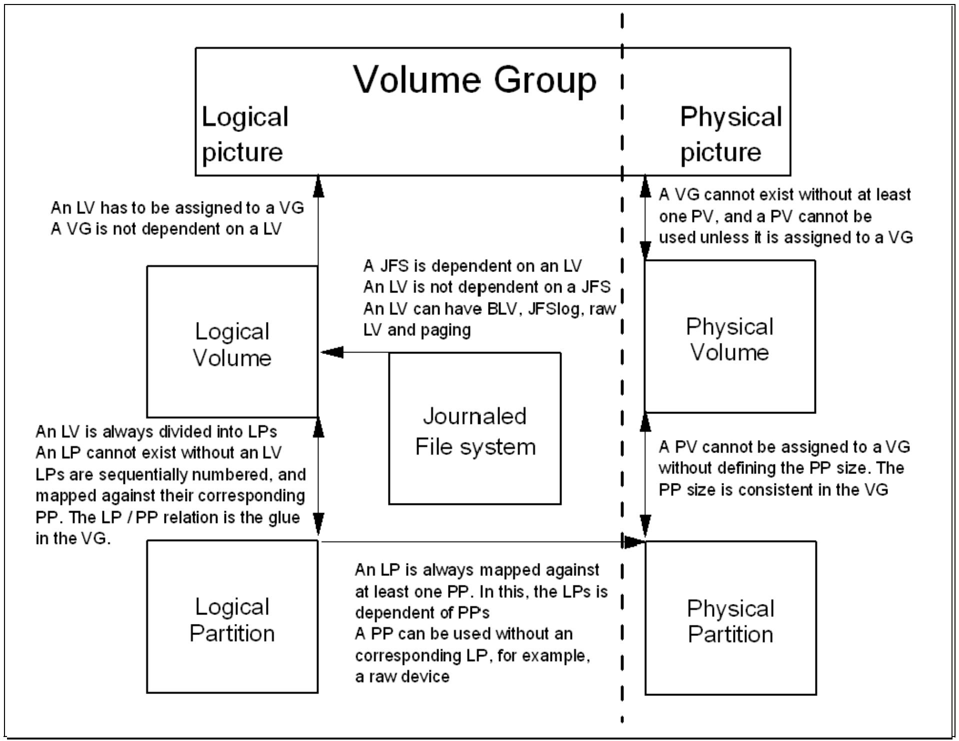

Figure 9. Dependencies in a volume group

Logical volumes can serve a number of system purposes, such as paging, but each logical volume that holds ordinary system data or 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 to be 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. Fragment size can be 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 might be 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 in which 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 in which a program is accessing 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 - 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 times minpgahead, the next for 4 times minpgahead, and so on until the number of pages reaches maxpgahead.

maxpgahead - 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 on the `vmtune` command is found in “The vmtune command” on page 177.

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, at which point no additional I/O is required. 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 remain in memory and do not get reused, the sync daemon writes them to disk, which might 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 normally `iostat`.

iostat

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. The `iostat` command generates reports that can be used to change 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 Version 4.3.2 and earlier, at each clock 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 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, 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 only mark 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 on the `iostat` command is found in “The iostat command” on page 215.

When finding 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.

filemon

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 on the `filemon` command in “The filemon command” on page 115.

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

fileplace

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 on the `fileplace` command in 5.4.2, “The fileplace command” on page 126.

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

lslv

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 on the `lslv` command is found in 5.7.3, “lslv” on page 132.

3.5 Network bound

When performance problems arise, your system might be totally innocent, while the real culprit 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 new high bandwidth interfaces like FIDDI and SOCC, 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 10 on page 71 , helps explain the layers involved in a read or write activity across TCP/IP and point out the network parameters used in each layer.

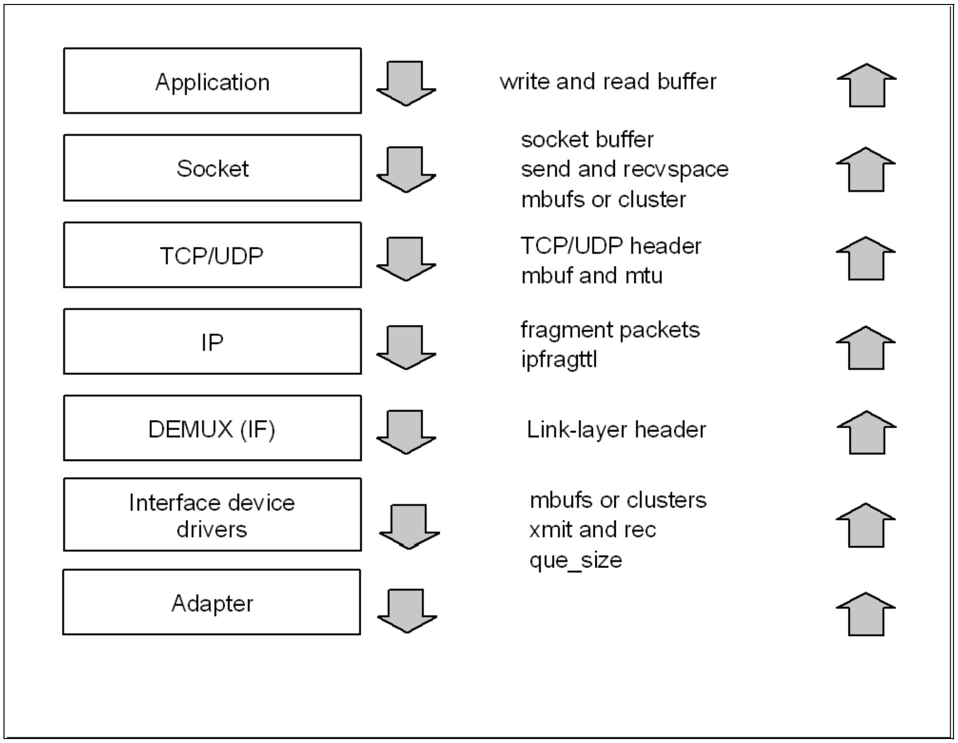

Figure 10. Network parameters

Chapter 6, “Network performance tools” on page 135, will discuss in more detail how to take into account mbufs and cluster, network packet tunables, adapter tunables for network performance tuning.

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

netstat

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. More on the `netstat` command in 6.4.3, “The netstat command” on page 142.

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

nfsstat

NFS gathers statistics on types of NFS operations performed, along with error information and performance indicators. You can use the `nfsstat` commands 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. You can also use this command to reinitialize this information. The `nfsstat` command splits its information into server and client parts. For example, use the command:

- `netstat -r` to see how application uses NFS

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

- `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 in 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 more information on the nfsstat command see *Performance Management Guide* and *Commands Reference - Volume 4*, SBOF-1877.

When going into more detailed output the `netpmon` command, using a trace facility, is useful.

netpmon

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 on the `netpmon` command in Chapter 6, “Network performance tools” on page 135.

3.6 Summary

The flowchart shown in starting this chapter is used in the summery, now with some suggestions included:

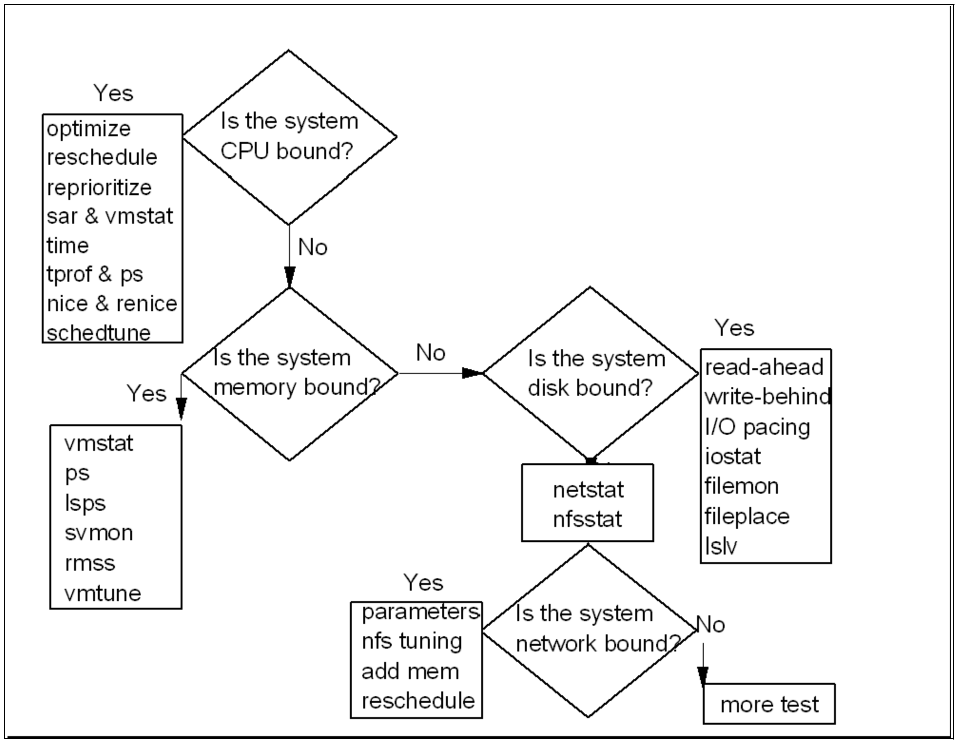

Figure 11. Performance tuning flowchart

3.7 Quiz

3.7.1 Quiz answers

Chapter 4. CPU performance tools

In this chapter is the following topics covered:

- The AIX scheduler
- The multiple run queue design of AIX Version 4.3.3
- The `nice` and `renice` commands
- The `schedtune` command

The scope of this chapter concentrates on the thread scheduling and the possibilities to manipulate the process priorities with the `schedtune` and the `nice` and `renice` command.

4.1 The AIX scheduler

The need of a scheduler is obvious. There are more threads 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 chose 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 seen in the left most column in the `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 is 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 3, “Performance tuning - getting started” on page 47.

In AIX Version 4, the five possible values for 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 value at each clock interrupt means that a thread may lose control because its priority value has risen above that of another dispatchable thread.

SCHED_FIFO2

The policy is the same as for SCHED_OTHER, except that it allows a thread which 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 beginning with operating system version 4.3.3.

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 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 satisfy the criterion that enables it to join the head of the run queue. This policy is only available beginning with operating system version 4.3.3.

4.1.1 Priority calculation on AIX versions prior to 4.3.2

The priority values differs from AIX versions previous to 4.3.2 and AIX versions 4.3.2 and later. Generally speaking, the lower the value is, the higher is the priority, with 0 as the lowest values and the greatest priority. In 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 (uses the CPU), the priority is recalculated and the value goes up, and by this the priority goes down. The longer a thread has existed without using CPU, the lower the value gets and by this, 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, and the thread running is released, and the thread from the run queue is given CPU time.

In Figure 12 on page 77 is the global run queue used at AIX versions prior to 4.3.2 symbolized. Thread A and thread B are released as thread C, thread D and thread E all have higher priority. When two or more threads have the same priority level, they will occupy consecutive positions in the run queue.

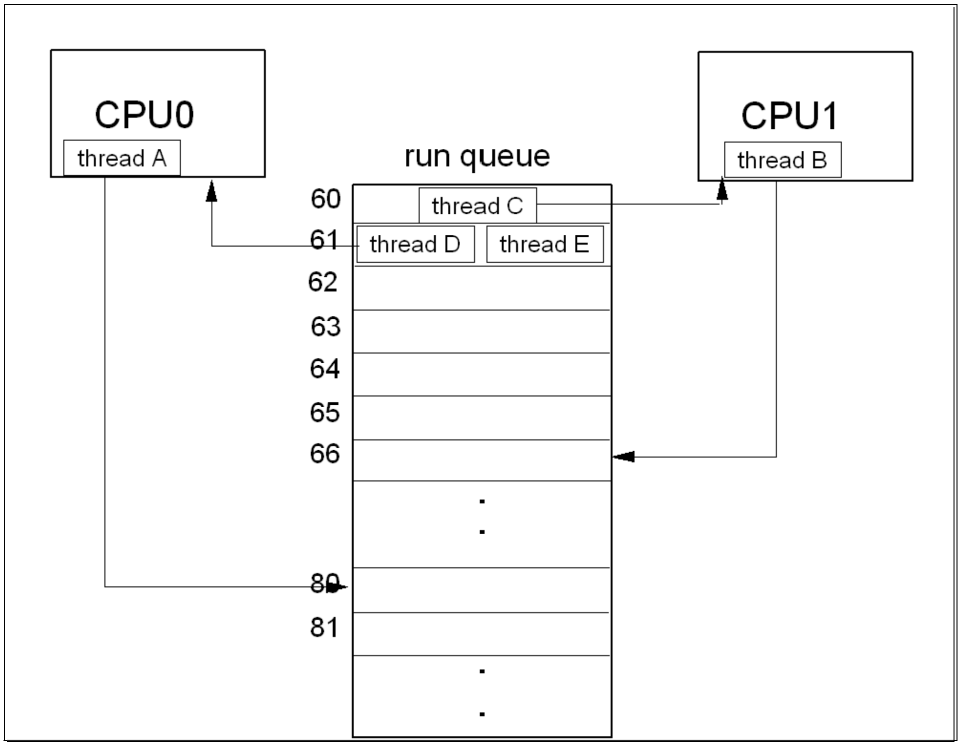

Figure 12. Run queue on AIX version prior to 4.3.3

Actually Figure 12 on page 77 is a simplification of the actual layout as shown in Figure 13 on page 78, with the following explaining text from *Performance Management Guide*:

> All the dispatchable threads of a given priority occupy consecutive positions in the run queue. Operating system 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.

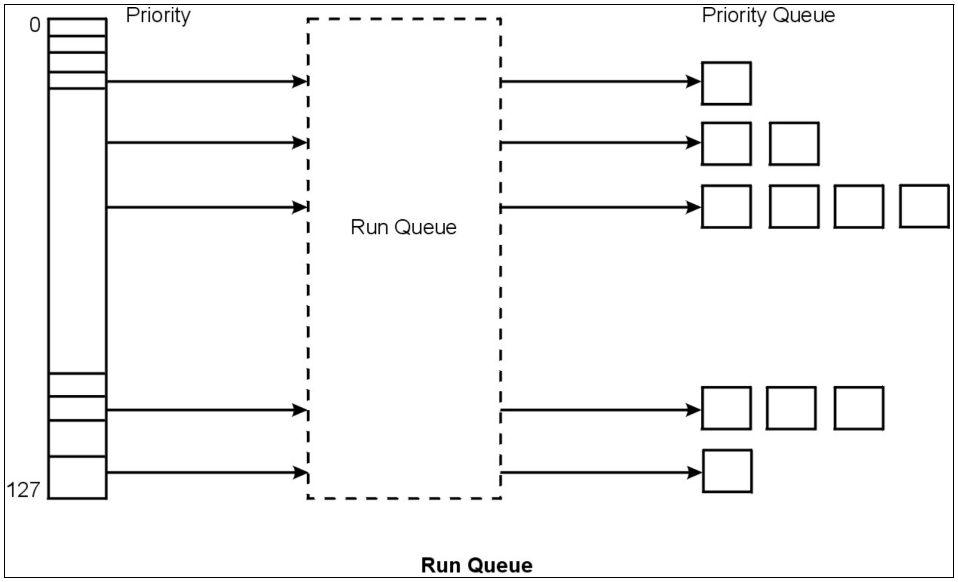

Figure 13. AIX Version 4, 128 run queues

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 (default 20 for foreground processes and 24 for background processes) and its CPU penalty.

One of the factors in the priority calculation is the *Recent CPU usage* value. One out of two calculation, defining the Recent CPU usage, follows:

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 has their recent CPU usage value recalculated and increased 100 times a second until reaching the maximum limit of 120. This value is show 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
```

This is not all to the recent CPU usage value. Once every second all threads, including those 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 that Recent CPU usage values for all processes ends up on 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 process and 24 for background processes.

From this you can see that 3 values can be manipulated. The nice value, and the SCHED_R, also called the weighting factor, and the SCHED_D, also called the decay factor.

4.1.2 Priority calculation on AIX version 4.3.2 and later

In this section, a couple of new definitions have come into consideration. 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 plus 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 user, also called base value (40), plus default nice value for a foreground process (20), in other words default 60.

The following calculation is used for 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

By this is meant: 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 the us as a constant factor. To get greater granularity in the run queue(s), the DEFAULT_NICE is set to 60. Let’s put in some values to show the calculation. The following values are just taken for the example.

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 = (2x 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

From this you can see that you still can manipulate 3 values. The nice value (as in the example), the SCHED_R, and the SCHED_D (for recent CPU usage). In the following sections are the multiple run queue layout and the commands which are used to change these values discussed.

4.2 Multiple run queues with load balancing in AIX Version 4.3.3

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 the run queue layout has changed. AIX Version 4.3.3 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.

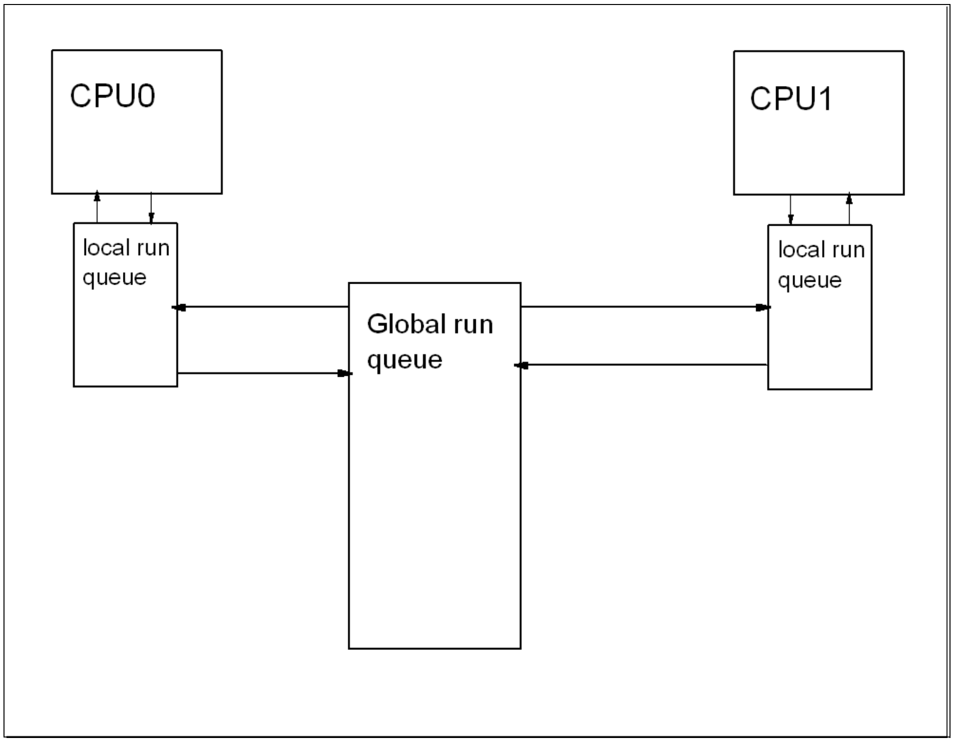

Figure 14. Run queue on AIX Version 4.3.3

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 courtesy 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.

4.2.1 Load balancing

In this scheduling design, load balancing is handled by a number of algorithms design to keep the various run queues of a system approximately equally utilized. there are four balancing algorithms:

Initial load balancing
Applies to newly created threads. When a 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 on the global queue.

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 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 has to 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 runned.
- Changes the threads run queue assignment and pointer.

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.

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.

4.3 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 options of the schedtune command: -r and -d. Each option specifies a parameter that is an integer from 0 through 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 SHED_D values are 16, as seen in the output.

4.3.1 Schedtune example 1

```
# /usr/samples/kernel/schedtune -r 0
```

(SCHED_R=0, SCHED_D=0.5) 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.

4.3.2 Schedtune example 2

```
# /usr/samples/kernel/schedtune -r 32 -d 32
```

(SCHED_R=1, SCHED_D=1) 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.

4.3.3 Schedtune example 3

The most likely reason to manipulate the values would be to make sure that background processes does 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
```

(R=0.15625, 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 15 on page 86 this is graphically shown. Because the CPU penalty will get a maximum value of 18, the foreground process with nice value 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:

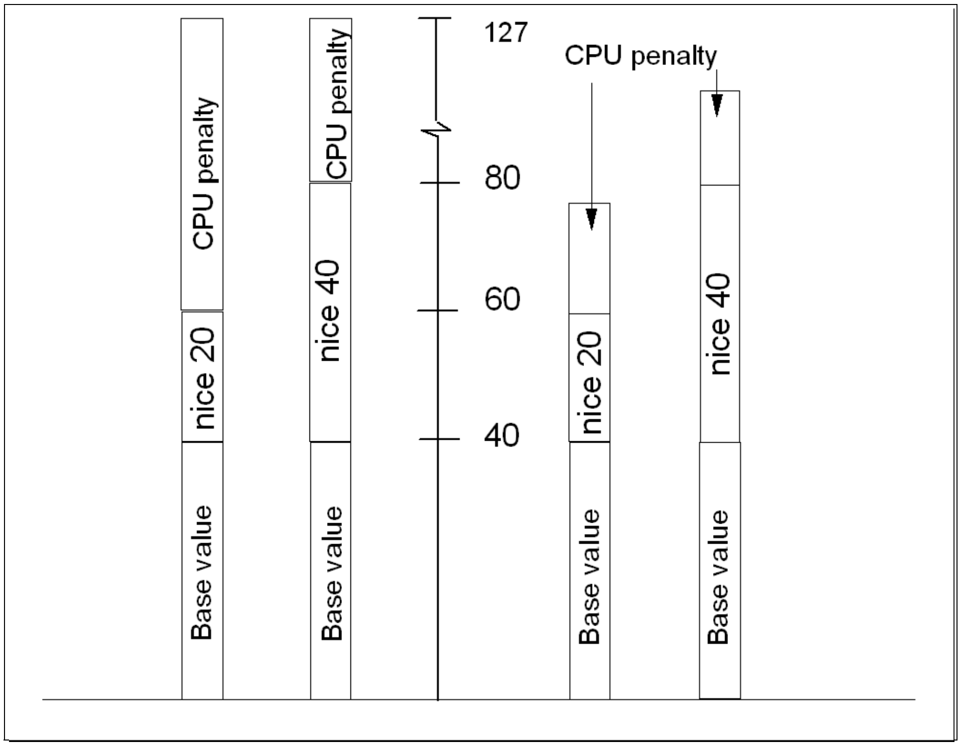

Figure 15. CPU penalty example

4.3.4 SCHED_R and SCHED_D guidelines

Here are some guidelines for SCHED_R and SCHED_D:

Table 9. Some useful schedtune flags

Flag	Description
-r	Manipulates the SCHED_R weighting factor
-d	Manipulates the SCHED_D decay factor
-D	Resets all schedtune values to default values

Smaller values of SCHED_R narrow the priority range and make the nice value more of an impact on the priority.

Larger values of SCED_R widen the priority range and make the nice value 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.

4.4 The nice and renice commands

The nice value has been explained in previous sections. The nice value can be seen with the `ps` command in the NI collumn:

```
$ 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 `tctestprg` ha been started in the background and the and the nice value is as seen 24, while the `ps` command running in the foreground has a nice value of 20. All outputs from the `ps` command has been edited in this section to fit the screen.

4.4.1 Running a command with nice

Any user can run a command at 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. In this case, 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.

For example, when being user thomasc, the nice values available are 1 to 19, but the maximum value is 39 (24 + 19 = 43).

```
# 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 - is only an option marker, and the other - 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
  200001A 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 execute the `nice` command, to get the same result as in the previous example, would be with the -n flag as follows:

```
#nice -n -15 ./tprof/tctestprg &
```

It is actually only in this case, where root lessens the nice value, a significant change is seen in the priority value. In the output above is the nice=39 and nice=24 close to even in the priority values (PRI column), but process 16304, started by root with nice 5 has a significant advantage with priority values in the range of 75 and 100 in this particular test case.

In the out put is also shown the scenario when a process is executed with a fixed priority (PID 16792). In the PRI collumn is the set priority shown - 59. And the NI collumn 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.

Table 10. Some nice flags

Flags	Description
-<increment>	Increments a command's priority up or down, by specifying a positive or negative number
-n<increment>	As above

4.4.2 Changing the nice value on a running thread

With the renice command, which has an similar syntax as the nice command, you can modify the nice value on a running process. The example from the previous section is used. Lets 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 point out which program (or more correctly - thread) is to be manipulated.

Table 11. Some renice flags

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

4.5 Summary

The conclusion of the chapter is:

Don’t manipulate the scheduler without a throughout understanding of the mechanisms controlling the scheduler.

4.5.1 The schedtune command

The `schedtune` command is used to manipulate the scheduler and the swapper. There are some major differences between in the commands between 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] } ]
```

In the following table are some, from a CPU tuning perspective, useful schedtune flags. For a more information on the `schedtune` command refer to *Performance Management Guide* (only available online at the time of publishing) and *Commands Reference - Volume 5*, SBOF-1877.

Table 12. Some useful schedtune flags

Flag	Description
-r	Manipulates the SCHED_R weighting factor
-d	Manipulates the SCHED_D decay factor
-D	Resets all schedtune values to default values

4.5.2 The nice and renice commands

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 useful nice flags:

Table 13. Some useful nice flags

Flags	Description
-<increment>	Increments a command's priority up or down, by specifying a positive or negative number
-n<increment>	As above

The syntax of the `renice` command is:

```
renice [ -n Increment ] [ -g | -p | -u ] ID ...
```

Some useful `renic`e flags:

Table 14. Some renice flags

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

For more information on the nice and renice commands refer to *Performance Management Guide* and C*ommands Reference - Volume 4,* SBOF-1877.

4.6 Quiz

4.6.1 Quiz answers

4.7 Exercise

1. Find a process with high recent 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 15 on page 86). Restore the default values.

4.8 The ps command

In the following chapter 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 is a useful tool to help you determine which processes are running and how much resources they use.

4.8.1 Use of ps command in CPU usage study

Three of the possible `ps` output columns report CPU usage, each in a different way.

Table 15. CPU related ps output

Column	Value
C	Recent used CPU time for process
TIME	Total CPU time sued 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 CPU dependence of the program

4.8.1.1 The C column

Let’s start with the **C** column. It can be generated by the `-l` and the `-f` flag. In this column is CPU utilization of process or thread 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. Large values indicate a CPU intensive process and result in lower process priority whereas small values indicate an I/O intensive process and result in a more favorable priority. In the following example is `tctestprog` running which is a CPU intensive program. The `vmstat` output shows that the CPU is used about 25% 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
```

Here comes the `ps` command handy. The following formatting sorts the output according to the third column with the biggest value at top, and shows only 15 lines from 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 example above you can tell that the tctestprog is the process with the most used CPU in recent time.

4.8.1.2 The TIME column

The second value mentioned is the **TIME** value. This value 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 event though the C column shows that the process gets a lot of CPU time, it is not yet in top on 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` is all processes that has been active since IPL, that is why they have accumulated more total TIME than the test program.

4.8.1.3 The %CPU column

Finally the **%CPU**. This column, 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 since 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%. In the example below 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 see that the test program, tctestprog, uses about 25% of available CPU resources since process start.

4.8.2 Use of ps command in memory usage study

The ps command also give useful information on memory usage. The most useful output is presented in the following columns:

Table 16. Memory related ps output

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

4.8.2.1 The SIZE column

The v flag generates the **SIZE** column. This is the virtual size (in paging space) in kilobytes of the data section of the process (displayed as SZ by other flags). This number is equal to the number of working segment pages of the process that have been touched times four. If some 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.

For 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
```

4.8.2.2 The RSS column

The v flag also produces the **RSS** column as seen 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 6th column, you will get the output accordingly to 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
```

4.8.2.3 The %MEM column

Finally the **%MEM** column, 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, like RSS, it tends the exaggerate the cost of a process that is sharing program text with other processes. Further, 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.

For example:

```
# 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
```

Finally you can combine all these column in 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 previous output are also these columns interesting:

PGIN

Number of page-ins caused by page faults. Since all I/O is classified as page faults, this is basically a measure of I/O volume.

TSIZ

Size of 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

Size of the resident set (real memory) of text. This is the number of code segment pages times 4. This number exaggerates memory use for programs of which 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.

4.9 The bindprocessor command

The bindprocessor command binds or unbinds the kernel threads of a process, or lists available processors. It uses two variables 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 which called fork. When the exec subroutine is called, thread properties are left unchanged.

To check what processors are available, do as follows:

```
# bindprocessor -q
The available processors are:  0 1 2 3
```

To bind process 16792 to processor 2, do as follows:

```
# bindprocessor 16792 2
```

To check what process threads are bound to what processor, check the BND collumn 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 -
(The output is edited to fit the screen)
```

4.10 The emstat command

The PowerPC architecture deleted 35 POWER instructions. To maintain compatibility with older binaries (which may contain these deleted instructions) the operating system 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 about how many instructions the system must emulate. The emulated instruction count should be used to determine whether an application needs to be recompiled to eliminate instructions that must be emulated on 601 PowerPC or 604 PowerPC processors. If an instruction has to 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 usually seen on PowerPC 604 systems. A typical example is a PowerPC 601 system that gets upgraded to a 604 system. If performance slows down instead of speeding up, 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 operating system version 4 is common architecture. However, the default architecture on operating system version 3 was for POWER, POWER2, and PowerPC 601. If these binaries ran on a PowerPC 604, some instructions could get emulated.

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

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

Note

In the future AIX release the `emstat` command is going to be moved to 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 that 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.

4.11 The tprof command

In this chapter the following topics will be covered:

Use of tprof for general CPU performance study

Use of tprof on a user program

4.11.1 Using 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 occurred, this information is used by the `tprof` command. The `tprof` command also records whether the process counter is in the kernel address space, the 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. This report 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 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 is a generic report 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` points out to the `tprof` command to run for 30 seconds

The `Total` collumn in the `__prof.all` is interesting. 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

Each tick is a 1/100 second. By this you can calculate the total amount of available ticks; about 30 seconds, times 100 ticks make a total of 3000 ticks. This according to the theory, but when looking at the output there are over 12000 total ticks. This is because the test system is a 4 way F50, so the available ticks are calculated in the following way:

Time (in seconds) x Number of available CPUs x 100

In the out put you see that both `tctestprg` used about 3200 ticks. Something around 25% of the total amount of available ticks. This is confirmed with a `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 is the total amount of ticks used by a specified type of process defined. Here is the ticks used by all three `wait` processes added together, and the two `tctestprg` are added together.

```
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
```

4.11.2 Using tprof on a program

The tprof command is also a useful tool for C or C++ or FORTRAN program that might be CPU-bound, in finding which sections of this program are most heavily using CPU. The `tprof` command specifies the user program to be profiled, executes the user program, and then produces a set of files containing reports. By this the output is divided down to sub-routine level.

This example is the most simple one, there are many more possibilities outside the scope of this subject:

```
# 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 will now be named __tctestprg.all, and the output is restricted to that 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
```

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


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
```

Chapter 5. LVM and JFS performance tools

The following topics are discussed in this chapter:

- Logical Volume Manager performance analysis using `lslv`.
- Journaled File system (JFS) performance analysis tools with `filemon` and `fileplace`.

All topics and tools discussed in this chapter will provide you with a *toolbox* of methods in determining logical volume manager, file system and disk I/O related performance issues.

5.1 Overview

In AIX operating system the handling of disk related I/O is based upon different AIX system levels as illustrated in Figure 16.

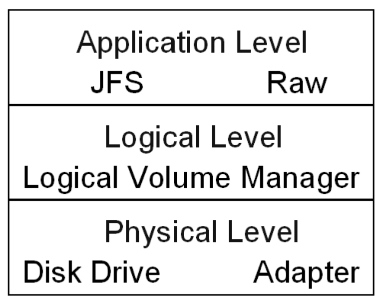

Figure 16. Disk, LVM and file system levels

The lowest level is the physical level, which are device drivers accessing the physical disks and using the corresponding adapters. On top of this 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 following performance analysis tools discussed will focus on the logical level (LVM) and on application level (JFS). The monitoring the physical level is primarily done by using the iostat command which is described in Chapter 7.1, “The iostat command” on page 215.

Covering the AIX Logical Volume Manager is a large subject and 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 4.3 System Management Guide: Operating System and Devices*, SC23-2525.

5.2 LVM performance analysis using lslv

There are various factors that affect a logical volume (LV) performance, for example the allocation position on the disk or the mirroring options. To get information about the logical volume use the LVM command `lslv`, 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

5.2.1 Logical volume attributes

To view logical volume attributes use the `lslv` command without any flags specified.

Example:

```
# 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
```

This 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 following attributes have to be taken into account:

COPIES	Indicate the number of physical copies. If copies equals 1 then the LV is un-mirrored. Values of 2 and 3 are used for mirrored LVs. The example used above uses a copy value of 2.
INTRA-POLICY	The intra-physical volume allocation policy specifies what strategy should be used for choosing physical partitions on a physical volume.
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.
SHED-POLICY	The two types of scheduling policies used for logical volumes with multiple copies are either sequential or parallel.
MWC	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 for every mirror 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.
BB POLICY	Flag specifying whether to use Bad Block Relocation, which redirects I/O requests from a bad disk block to a good 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.

5.2.1.1 Mirroring

To enhance the availability of a logical volume AIX supports the mirroring of logical volumes, by providing multiple copies of the logical volumes on different disks.

When using mirroring the write scheduling policies are:

Sequential	The sequential write policy waits for the write operation to complete for the previous physical partition before starting next write operation.
Parallel	The parallel write policy starts the write operation for all the physical partitions of a logical partition at the same time. The write returns when the slowest write operation is completed.

The parallel write scheduling policy gives the best performance and should be preferred when creating the mirrored LV.

There is in general following recommendation give the highest LVM availability:

- Use 3 logical partition copies (mirror twice) and include at least 3 physical volumes.
- Write verify switched on.
- Inter disk policy set to minimum, which means that the mirroring copies equals the number of physical volumes.
- Disk allocation policy set to strict, which means no LP copies on the same disk.
- Finally to fully enhance the availability the copies on the physical volumes attached to separate busses, adapters and power supplies.

Providing the highest availability has a negative impact on LVM performance, hence not all settings may be followed depending on the requirements.

5.2.1.2 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 17.

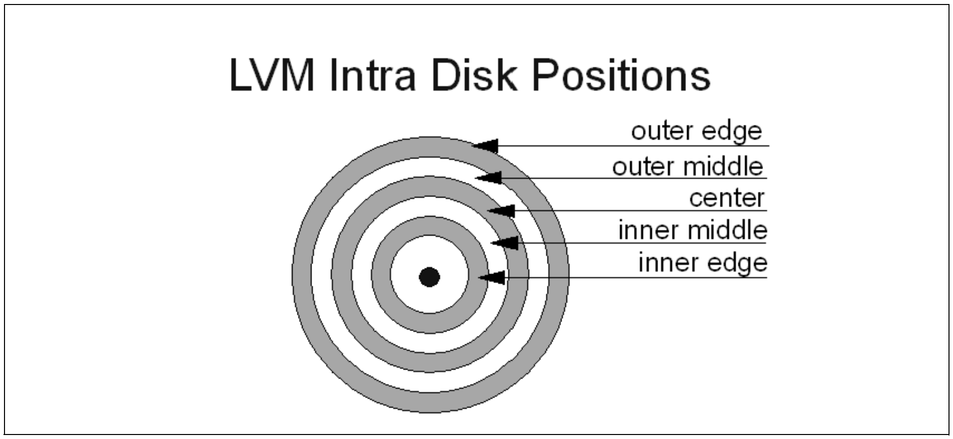

Figure 17. LVM intra disk positons

In terms of performance the following applies to the intra disk policies:

- The center allocation policy has the fastest average seek times.
- The outer middle and inner middle allocation policies provide reasonably good average seek times. This is the default setting when creating a new logical volume.
- The outer edge and inner edge policies have the slowest average seek times.

5.2.1.3 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 as few disks as possible. The MINIMUM policy provides the best availability.

The MAXIMUM inter disk allocation policy allocate 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 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 are used to hold all the physical partitions.

5.2.1.4 Striping

Striping is designed to increase the read/write performance of frequently accessed, large sequential files. 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, hence the old conclusion that striping does not provide availability, because the lack of mirroring is no longer valid. For more information on this issue see the *AIX Version 4.3 Differences Guide*, SG24-2014 (second edition).

When a striped logical volume is defined then two additional LV attributes are displayed by the lslv command:

STRIPE WIDTH Number of stripes

STRIPE SIZE 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.

5.2.2 Logical volume fragmentation

To check a logical volume for possible fragmentation, use the lslv flag -l.

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 shows that two disks are used (hdisk1 and hdisk2).

The COPIES column shows that the total number if logical partitions (LP) is 120, and since it is a mirrored LV both disk have the same amount of physical partitions PPs allocated (240 in total).

The IN BAND column shows the level of intra allocation policy in percent. 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% on hdisk1 and 90% on hdisk2 could follow this allocation request.

The DISTRIBUTION column shows how the physical partitions are allocated in each section of the intra policy. That is:

```
(outer edge) : (outer middle) : (center) : (inner middle) : (inner edge)
```

In this example the hdisk1 has allocated 83 PPs on the requested inner middle section and 37 on the outer edge. The hdisk2 allocates the intra policy request a better, by 25 block, hence the higher IN BAND level.

5.2.3 Logical volume allocation

To see the logical volume allocation of placement on the physical volume, use the 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 which represent: outer edge, outer middle, center, inner middle, and inner edge.

Each physical partitions is marked with either a number or a keyword, which are 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 above 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, whereas the LPs 84-120 are allocated in 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.

5.2.4 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 requires better performance, the availability in some cases might be impacted.

- No mirroring, which means the number of copies equals one (1)
- If mirroring is required then:
  - Write scheduling policy = parallel
  - Allocation policy = strict, which means each copy 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 operation are spread among physical volumes.

Additional performance improvement can be gained by creating a striped logical volume.

5.3 LVM and file system monitoring

To provide a more complete analysis of file system performance AIX Version 4.3 provides a monitoring utility filemon. This provides information of specific application or system I/O activity, assisting the performance problem determination process.

5.3.1 The filemon command

The `filemon` command monitors and presents trace data on the following four levels of file system utilization:

Logical file system — The `filemon` command monitors logical I/O operations on logical files. 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 or not the files

	are already buffered in memory. I/O statistics are kept on a per-file basis.
Virtual memory system	The `filemon` command monitors physical I/O operations (that is, paging) between segments and their images on disk. I/O statistics are kept on a per-segment basis.
Logical volumes	The `filemon` command monitors I/O operations on logical volumes. I/O statistics are kept on a per-logical-volume basis.
Physical volumes	The `filemon` command monitors I/O operations on physical volumes. At this level, physical resource utilizations are obtained. I/O statistics are kept on a per-physical-volume basis.

5.3.1.1 Using the filemon program

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 as root and `filemon` cannot be executed in parallel with other trace-based commands like `tprof` and `netpmon`.

Tracing is started implicitly by the `filemon` command, but the trace can be controlled by the normal trace utilities: `trcstop`, `trcoff`, `trcon`.

When tracing is stopped with `trcstop` filemon writes a report either to stdout or a specified file.

To specify the levels of data collected on all the layers, or on specific layers by specifying the -O layer option. The default is to collect data on the VM segments, LVM, and physical layers. Both summary and detailed reports are 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 gives a simple example of filemon usage. Example:

```
# 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 program is started with the flag -O specifying only the logical filesystem (lf) data 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 is in this example placed in a dedicated file using the -o flag. The default is to write the report to the standard output.

5.3.2 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* which includes lv, pv, and vm.

In the following we will take a closer look on the filemon output reports using the example from Chapter 5.3.1, “The filemon command” on page 115.

5.3.2.1 Logical file level report

The logical file level report as shown below provides two sections, the *Most Active Files Report* 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, which is:

#MBS	Total number of megabytes transferred to/from file. The rows are sorted by this field, in decreasing order.
#opns	Number of times the file was opened during 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 detailed report).
volume:inode	Name of volume that contains the file, and the file's i-node 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 used shows that the file created testMirrorFile is the most active, with the 1 MB read and 1 MB write operations. Notice the read and write operations are made per 512 bytes. This shows that the dd command used in the example used 512 byte block size per default. The zero and null files do not have inodes as they are not connected to any file system, but special device files.

The filemon file level report it is very evident to see which files are generating the most I/O demand.

The *Detailed File Stats Report* provides information every 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	I-node 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/sdev), in bytes.

read times (msec) The read response-time statistics (avg/min/max/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 filemon example above, is again focusing on the file testMirrorFile. Here the read size and write size discovered above of 512 bytes is even more evident. As it is the only read/write size used all the same, so the standard deviation sdev becomes 0. Interesting in the detailed file statistics report is the read/write times. These can show, among other things, how good the file system cache is performing.

5.3.2.2 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 for detailed statistics on the individual LVs.

The logical volume level report was generated with the same example as in Chapter 5.3.1.1, “Using the filemon program” on page 116, except the output leve -O lv was used.

```
# filemon -o /tmp/filemonLF.out -O lv
```

Following is an extraction of 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
...
```

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.

The logical volume level report of the filemon example shows clearly that the mirrlv is the most utilized LV. The report shows that 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.

5.3.2.3 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 for detailed statistics on the individual PVs.

The physical volume level report was generated with the same example as in Chapter 5.3.1.1, “Using the filemon program” on page 116, except the output leve -O pv was used.

```
# filemon -o /tmp/filemonLF.out -O pv
```

Following is an extraction of 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
...
```

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 of the mirrored copies of mirrlv. Notice that hdisk1 has a slightly higher write block size than hdisk2 (and due to that probably also a slightly higher throughput).

This is because the jfslog loglv00 is located on hdisk1.

5.3.2.4 Virtual Memory level report

The virtual memory level report provides two sections, the *Most Active Segments* report and the *Detailed VM Segment Stats* report for detailed statistics.

The virtual memory level report was generated with the same example as in Chapter 5.3.1.1, “Using the filemon program” on page 116, except the output leve -O vmwas used.

```
# filemon -o /tmp/filemonLF.out -O vm
```

Following is an extraction of 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
...
```

#MBs Total number of megabytes transferred to/from 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).

#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, name of 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.

5.3.3 Typical AIX system behavior

When using the filemon program for performance analysis 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 always very active. Because files and directories access permissions are 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 and/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 seq. lengths 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. You have to further investigate with the `fileplace` command, see Chapter 5.7.2, “fileplace” on page 131.

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 file systems.
- If the workloads access pattern is predominantly random, you may want to consider adding disks and distributing the randomly accessed files across more drives.
- If the workloads 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.

5.4 File System Performance

There 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 lets look at how the AIX Journaled File System (JFS) is organized.

5.4.1 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 byte (4 KB). The JFS i-node contains a information structure of the file together with an array of 8 pointers to data blocks. A file which is less then 32 KB is referenced directly from the i-node.

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 a file size of 1024x4 KB=4 MB is possible.

For larger files then 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 512*1024*4 KB=2 GB files. The Figure 18 on page 125 illustrates the addressing using double indirection.

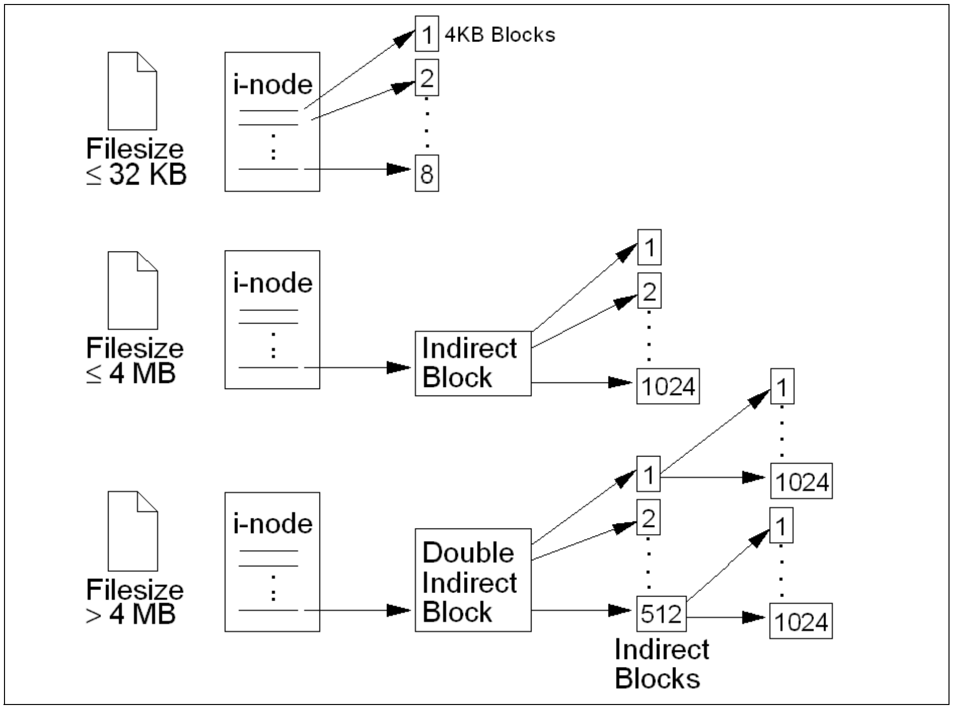

Figure 18. JFS file system organization.

Since AIX V4.2 support for even larger files was added, by defining a new type of JFS file system the *bigfile* file system. In the bigfile file system the double indirect are using references to 128 KB blocks rather then 4 KB blocks. However the first indirect block still points to 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.

5.4.2 The fileplace command

The usage 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 result in a large number of seeks and longer I/O response time. At some point, the system administrator may decide reorganizing the placement of files within 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.

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 is displaying the logical fragmentation of a large file sixMB (which also happens to be the file size). The general information displayed by filemon is:

File	Name of the file
Size	Filesize in bytes
Vol	Name of the logical volume of the file system
Blk Size	Physical block size 4 KB

Frag Size	Fragment size, typical also 4 KB, but can be specified to values 512, 1 KB, 2 KB at file system creation time.
Nfrags	The total amount of fragments used by the file.
Compress	Compression of file system, per default no.
Inode	The inode reference number.
Mode/Owner/Group	General UNIX file system level i-node information

This file has due to its size both indirect blocks (75321, 77001) and a double indirect blocks (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 filemon -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 the filemon -p flag the physical block numbers and physical volume or volumes are shown.

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.

Note

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.

5.4.3 File system de-fragmentation

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 new 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. The `defragfs` command is intended for fragmented and compressed file systems. However, you can use the `defragfs` command to increase contiguous free space in non fragmented file systems.

Another simple way of reorganizing the file system is to re-create the file system using a backup of the file system.

5.5 General recommendations on I/O performance

By using `lslv`, `fileplace`, `filemon`, and `iostat`, you can identify I/O, volume group, and logical volume problems. Following are general recommendation on how to achieve good LVM and file system performance and when to use the tools described in this chapter.

Logical volume organization for highest performance:

- Allocate hot LVs to different PVs to reduce disk contention.
- Spread hot LV across multiple PVs so that parallel access is possible.
- Place hottest LVs in center of PVs, moderate LVs in the middle of PVs, and place coldest LVs on edges of PVs so that the hottest logical volumes have the fastest access time.

- Do not use mirroring if possible since mirroring performs multiple writes that will affect performance in writing, and it only provides marginal improvement in reading.
  - If mirroring is needed, set scheduling policy to parallel and allocation policy to strict. Parallel scheduling policy will enable reading from closest disk, and strict allocation policy allocates each copy on separate PVs.
- Make 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 seek time.
- Set write verify to no so that there is no follow-up read (similar to parity check) performed following a write.

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.

Striping recommendations:

- The stripe unit size should be equal to the max_coalesce, which is by default 64 KB. The max_coalesce value is the largest request size (in terms of data transmitted) that the SCSI device driver will build.
- Use a minpghead value of 2: This will give the number of minimum pages with which sequential read-ahead starts.
- Using a maxpgahead of 16 times the number of disk drives causes the maximum pages to be read ahead to be done in units of the stripe size (64 KB) times the number of disk drives, resulting in the reading of one stripe unit from each disk for each read ahead. If possible, modify applications that use striped logical volumes to perform I/O in units of 64 KB.
- Limitations of striping
  - Mirroring with striping prior to AIX Version 4.3.3 is not possible. On AIX 4.3.3 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.

File system related 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.

`lslv` usage scenario:

- Can hot file systems be better located on physical drive or be spread across multiple physical drives?

`filemon` usage scenarios:

- Are hot files local or remote?
- Does paging space dominate disk utilization?
- Look for heavy physical volume utilization. Is the type of drive (SCSI-1, SCSI-2, tape, and so on) or SCSI adapter causing a bottleneck?

`fileplace` usage scenarios:

- Does the application perform a lot of synchronous (non-cached) file I/O?
- Look for file fragmentation. Are hot files heavily fragmented?

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

5.6 Overhead of using performing tools.

As in any performance measurement on a system, each measurement consumes some resources which is referred to as the *overhead* of the performance tools. The following information is take from the *AIX Performance Tuning Guide, Versions 3.2 and 4*, SC23-2365:

`lslv` This command uses mainly 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.

Please note that the computing time measurement are from an old RS/6000 Model 320.

5.7 Commands summary

For a complete reference of the following command use the *AIX Version 4.3 Command Reference* or the online man pages.

5.7.1 filemon

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 ]
```

Table 17. Commonly used flags of the filemon comman

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), 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.

5.7.2 fileplace

The `fileplace` command displays the placement of file blocks within logical or physical volumes. The command has the following syntax:

```
fileplace [ {-l | -p } [-i ] [-v ] ] File
```

Table 18. Commonly used flags of the fileplace comman

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.
-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.

5.7.3 lslv

The `lslv` command displays information about a logical volume. The command has the following syntax:

Display Logical Volume Information:

```
lslv [ -L ] [ -l|-m ] [ -nPhysicalVolume ] LogicalVolume
```

Display Logical Volume Allocation Map:

```
lslv [ -L ] [ -nPhysicalVolume ] -pPhysicalVolume [ LogicalVolume ]
```

Table 19. Commonly used flags of the filemon comman

Flag	Description
-l	Lists the following 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.

5.8 Quiz

5.8.1 Answers

5.9 Exercises

The following exercises provide the setting for additional learning.

1. On an test system with preferably two spare disks create a testvg volume group. Create test logical volumes with the different parameters dicussed in this chapter: mirrored LV, intra disk policy, inter disk policy, strict policy, 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 exisiting file system create a large file. Verify its fragmentation as well as space efficiency and sequentially with the filemon command.

Chapter 6. Network performance tools

The following topics are discussed in this chapter:

- Network performance problems overview
- Network monitoring tools
- Network tuning tools

This chapter deals with network performance problems. It describes network examination and problem solving procedures.

6.1 Overview

The first aid tools for network performance are the `ping` and the `netstat` commands. Usually they give you enough informations to discover your problems, if not, read this chapter to know more about network performance issue.

To understand the performance characteristics of network subsystem in AIX, you must first understand some of the underlying architecture. The Figure 19 on page 137 shows the path of data from an application in one system to another application in a remote system. Following the diagram:

- As an application writes to a socket, the data is copied from user space into the socket send buffer in 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 sizes of the buffers in system virtual memory that are used by the input is limited by 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,
  - 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 is given 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 had 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 `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.

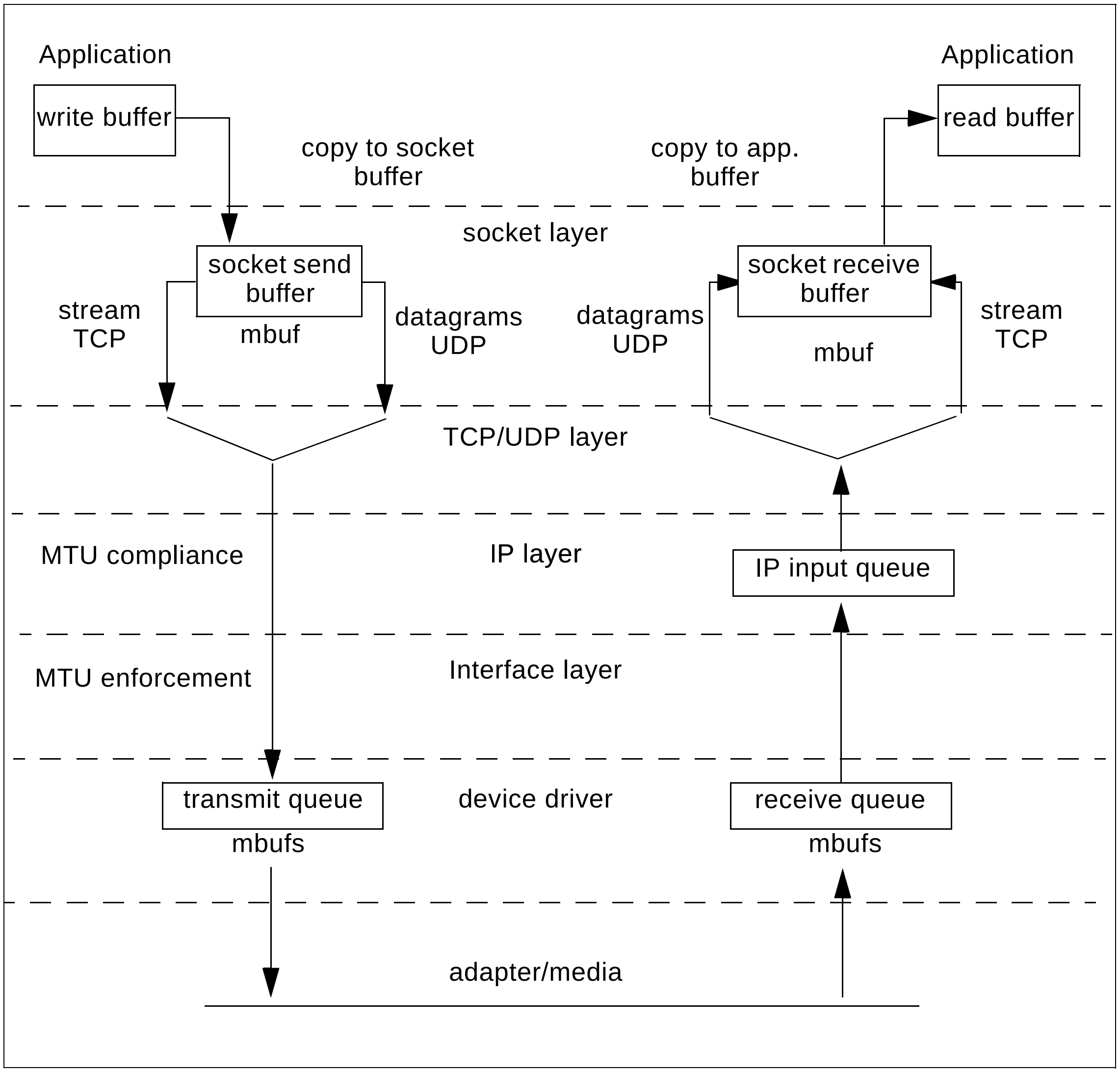

Figure 19. UDP/TCP/IP data flow

6.2 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 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 follow the procedure:

Bring down the interface:

```
# ifconfig en0 detach
```

Change value of the appropriate parameter:

```
# chdev -l ent0 -a tx_que_size=128
ent0 changed
```

Bring interface back to the up state:

```
# ifconfig en0 up
```

To check if queues size should be change 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. 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 says 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 requires the same actions as would be needed if the Max Packets on S/W Transmit Queue reaches the tx_que_size. The transmit queue size has to be increased.

6.3 Protocols tuning

The main goal in network performance issue is to balance demands of users against resource constraints to ensure acceptable network performance.

You can do this in following steps:

- Characterize workload, configuration and bandwidth
- Measure performance:
  - Run tools, identify bottlenecks
  - Tune network parameters
- Monitoring tools
  - `netstat`, `tcpdump`, `iptrace`
- Tuning tools
  - `no`, `nfso`, `chdev`, `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 the 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 parameter which you can tune for this purpose:

- thewall — Kernel variable, that 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 application. Default value for tcp_sendspace is16384.
- tcp_recvspace — Kernel variable used as the default socket receive buffer size when an application opens TCP socket. Default value for tcp_recvspace is 16384.
- udp_sendspace — Kernel variable, that 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. Default value for udp_sendspace is 9216.
- udp_recvspace — Kernel variable, that sets the size limit of the receive space buffer for any single UDP socket. Default value for udp_recvspace is 41920.
- rfc1323 — If the value of this variable is non-zero, it alows the TCP window size to maximum of 32 bits insted of 16 bits. What this means, is that you can set tcp_recvspace and tcp_sendspace to be grather than 64Kb.
- sb_max — Kernel variable, that controls the upper limit for any buffers.
- ipqmaxlen — Kernel variable, that 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 penantly for insufficient queue length are dropped packet.

Note

The values of the tcp/udp_sendspace and tcp/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

The network application, like a backup over the network, that sends data in a big chunks, can generate socket buffer overflows. The 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 can not be bigger than socket buffer space. To increase performance of such a 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. value.

6.4 Network monitoring tools

This section describes the most common network monitoring tools.

6.4.1 The vmstat command

You should invoke network monitoring tools in order to get more statistics for isolation when you suspect have a network bottleneck. When the `vmstat` command shows significant amount of idle time that does not fit the problem, the system may be network bounded.

```
# 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 nondisk wait is in the `id` column. Nondisk wait either include network I/O wait or terminal I/O wait. If it is no terminal I/O wait than system is waiting for network I/O to complete. You should run one of the network monitoring tools to find out the reason for network I/O wait.

6.4.2 The ping command

When you have connection problem the first tool which 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:

```
# 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 simple test checks round-trip times and packet loss statistics

6.4.3 The netstat command

The most common network monitoring tool is `netstat`. The `netstat` is used to show network status. It gives you a good indication of the reliability of the local network interface. Traditionally, it is used more for problem determination than for performance measurement. However, it is useful in determining the amount of traffic on the network, to ascertain 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 enables more extensive statistics for network memory services (for AIX 4.3.2 and latter), you should set kernel variable `extendednetstats` to 1 first:

```
# 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 very important statistics. In this example are 4 Kbytes allocated out of a possible limit 16 Mbytes. This limit can be regulated by thewall kernel variable. The second important 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 protocols statistic use the `netstat` command with the -p flag and the appropriate protocol name. To have statistic for IP protocol, enter:

```
# 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
```

The when the ipintrq overflows couter has nonzero value you should change length of the IP input queue using the `no` command:

```
# no -o ipqmaxlen=100
```

The `netstat` command is also useful in determining the amount of traffic on the network, to determine whether performance problems are due to congestion. To check the amount of packets that passing interfaces and number of input/output errors, enter:

```
# 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
```

6.4.4 The netpmon command

The `netpmon` is the tool used for network I/O analyze. It usee `trace` as means to collect statistics about events occurring in the network code in the kernel. Tracing must be stopped using a `trcstop` command. The `netpmon` then generates all the 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 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 NFS work load during the `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 interrupt handler executed on behalf of network-related events. For other statistic 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`.

6.4.5 The tcpdump and iptrace commands

Tools discussed previous allow you see various number of statistics and network-type events in the AIX kernel. However, sometimes you get a problem where statistic counters are not enough to find the cause of the problem. Sometimes you need to see the real data “crossing the wire”. There are two commands that let you see every incoming and outgonig 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 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 says 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 an `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 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 11th 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 12th 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 query from host 9.3.240.59 to DNS server 9.3.240.2. The output from the `nslookup` command is shown below:

```
# 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 as shown below:

```
# 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:

```
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. Every packet is divided into a few parts. Each part describes different network protocol level. There is token ring (TOK), IP, UDP and application (DNS) part. The first packet is send by host 9.3.240.59 and this is query about IP address of www.prokom.pl host. The second one is the answer.

6.5 Network tuning 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 check what system parameters can you change and what is the current value of every parameter, enter:

```
# 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 value of the `thewall` system parameter, shown as 18420, to 36840, enter:

```
# no -o thewall=36840
```

The `ifconfig` command can be used to assign an address to a network interface or to configure/display the current configuration information. For tuning purposes, it is used to change the mtu size:

```
# ifconfig en0 mtu 1024
```

Note

The mtu parameter 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 ODM database. To display current value of the parameters of the `en0` interface use the `lsattr` command:

```
# 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 change vale of the mtu parameter, enter:

```
# chdev -l en0 -a mtu=1024
en0 changed
```

6.6 Name resolution

If a network connection seems inexplicably slow sometimes but all right at other times, it is good idea to check name resolution configuration for your system. Do 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 remote DNS server or remote NIS server. So, if one of them is down, you have to wait until aTCP time-out occur. The name can be resolved by alternate source, which can be 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 /etc/resolve.conf file for IP address of name server and try to `ping` it. If you can `ping` it, then it is up and reachable. If not, try to play with different name resolution ordering.

6.7 NFS tuning

This section discuss the NFS server and NFS client performance issue.

6.7.1 NFS server performance

When narrowing down the performance discussion on servers to NFS specifics, the issue is often related to dropped packages. NFS servers may sometimes drop packets due to overload.

One common place where a server will drop packets is the UDP socket buffer. Remember that 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 use of 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 testsystem the buffer size was sufficent)
```

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 sockets attached to the NFS port and all incoming data is buffered on those ports. The default size of this buffer is 60000 bytes. Doing some quick math dividing that number by the size of the default NFS Version 3 write packet (32765) you find that it will take only 2 simultaneous write packets to overflow that buffer. That could be done by just one NFS client (with the default configurations). Practically speaking, however, it is not as easy as it sounds to overflow the buffer. As soon as the first packet hits the socket, an nfsd will be awakened to start taking the data off.

So one of two things has to happen. There has to be high volume, or high burst traffic on the socket. If there is high volume, a mixture of lots of writes plus other possibly non-write NFS traffic, there may not be enough nfsds to take the data off the socket fast enough to keep up with the volume (recall that 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 handling. In the case of high volume, it may be sufficient to just increase the number of nfsds running on the system. Since there is no significant penalty for running with more nfsds 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, modifies the SRC database code to reflect the new number, and restart the daemons indicated

In the case of high burst 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 nfsds time catch up with the burst. Memory dedicated to this socket will not be available for any other use so it must be noted that making the socket larger may result in that memory 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. To check the NFS kernel options, use the `nfso` command:

```
# nfso -a
portcheck= 0
udpchecksum= 1
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
```

If you change the *nfsbuffer* sizes, you must verify that the kernel variable *sb_max* is greater then the NFS buffer values chosen. The default value of sb_max is 1048576 on AIX Version 4.3.3. If you need to increase the sb_max value. This can be done with the `no` command. Remember that everything changed with `no` or `nfso` is valid only until next boot, if these changes have been added to some bootup script, for example /etc/rc.nfs.

6.7.2 NFS client performance considerations

The NFS client performance discussion often concentrates on the number of biods used. For biod daemons, there is a default number of biods (six for a NFS V2 mount, four for a NFS V3 mount) that may operate on any one remote mounted file system at one time. The idea behind this limitation is that allowing more than a set number of biods to operate against the server at one time may over load the server. Since this is configurable on a per-mount basis on the client, adjustments can be made to configure client mounts by the server capabilities.

When evaluating how many biods 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 process 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.

Since up to six biods 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 6 biods. 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 biods could actually decrease the response dramatically due to the server dropping packets and resulting in time-outs and retransmits.

There are also some mount options that may improve the performance on the client. 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 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
```

6.7.3 Mount options

The mount command has several NFS specific options that may affect performance.

Slow mounts from AIX Version 4.2.1 or later clients running NFS V3 to AIX Version 4.1.5 or earlier and other non-AIX servers running NFS V2. NFS V3 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 V3 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 V2 for the mount request:

```
# mount -o proto=udp,vers=2,soft,retry=1 server4:/tmp /mnt
```

6.8 Commands

For a complete reference of the following command use the *AIX Version 4.3 Command Reference* or the online man pages.

6.8.1 netstat

The syntax of the `netstat` command is:

```
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 Subsytem

/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 point of view:

Table 20. Some useful netstat flags

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

6.8.2 tcpdump

The syntax of the `iptrace` command is:

```
tcpdump [ -I ] [ -n ] [ -N ] [ -t ] [ -v ] [ -c Count ] [ -i Interface ] [
-w File ][ Expression ]
```

Some useful `iptrace` flags:

Table 21. Some useful tcpdump commands

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*

6.8.3 iptrace

The syntax of the `iptrace` command is:

```
iptrace [ -a ] [ -e ] [ -PProtocol ] [ -iInterface ] [ -pPort ] [ -sHost [
-b ] ] [ -dHost [ -b ] ] LogFile
```

Some useful `iptrace` flags:

Table 22. Some useful iptrace commands

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

6.8.4 ipreport

The syntax of the `ipreport` command is:

```
ipreport [ -e ] [ -r ] [ -n ] [ -s ] LogFile
```

Some useful `ipreport` flags:

Table 23. Some useful ipreport flags

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

6.9 References

The following publications contain more information about network tuning procedures.

- *AIX Versions 3.2 and 4 Performance Tuning Guide*,
- *AIX Version 4.3 Commands Reference, Volume 3*, SC23-4117
- *AIX Version 4.3 Commands Reference, Volume 4*, SC23-4118
- *IBM Certification Study Guide AIX V 4.3 Communication*, SG24-6186

6.10 Quiz

6.10.1 Answers

6.11 Exercises

1. Catch a telnet session using the `tcpdump` command.
2. Catch 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 using the `netstat` command with `-p` flag.

6.12 Collecting data using the sar command

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

The `sar` command is one of the first performance monitors that will be used by the administrator, however it is only one tool in a variety of tools and should not be used as the definitive performance tool. 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.

Examples

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 and in this example it uses the default as shown 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 without any flags but using interval and number flags will have the output as shown below. This is the same as running the `sar -u 1 10` command.

```
# 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. 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 status of the process, kernel-thread, i-node, 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 the output it is an indication of what the main flag options can do. When running the `sar` command a combination of the flags can be used to get the output required for analyses for 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
```

6.12.1 The sar command

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

The `sar` command syntax is as follows:

sar [{ -A | [-a] [-b] [-c] [-k] [-m] [-q] [-r] [-u] [-v] [-w] [-y] }] [-P *ProcessorIdentifier*, ... | ALL] [-e*hh* [:*mm* [:*ss*]]] [-f*File*] [-i*Seconds*] [-o*File*] [-s*hh* [:*mm* [:*ss*]]] [*Interval* [*Number*]]

The accounting system, based on the values in the *Number* and *Interval* parameters, writes information the specified number of times spaced at the specified intervals in seconds. 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 waiting for disk I/O. The I/O may be required 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 process 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.

Table 24. The sar command flags

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* Calls to any of several i-node lookup routines that support multiple file system types. The iget routines return a pointer to the i-node 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 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.
`-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.

Flag	Description
-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 which 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.
-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.

Flag	Description
`-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 occurance.
`-v`	Reports status of the process, kernel-thread, i-node, 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 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.

6.12.2 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 a specified number of times (*`Number`*) at a specified interval measured in seconds (*`Interval`*). 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.

The operating system 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

6.12.3 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/sadd` 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 *Number* times at Interval 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/sardd` 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
```

6.13 Quiz

6.13.1 Answers

6.14 Exercises

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.

6.15 topas command

The `topas` command reports vital statistics about the activity on the local system in a character terminal. It requires the operating system version 4.3.3 `perfagent.tools` fileset to be installed on the system. This command is a kind of compilation of others diagnostic commands like a `sar`, `vmstat`, `iostat` and `netstat`. Its allows you to see all of this statistics on one screen so it is easy to observe cross-connection between them. As you can see, in the Figure 20 on page 173, the command output is divided into five subsections of statistics:

- EVENTS/QUEUES — Displays the per-second frequency of selected system-global events and the average size of the thread run- and wait queues:
- FILE/TTY — Displays the per-second frequency of selected file and tty statistics.
- PAGING — Displays the per-second frequency of paging statistics.
- MEMORY — Displays the real memory size and the distribution of memory in use.
- PAGING SPACE — Display size and utilization of paging space.

```
Topas Monitor for host:    client1              EVENTS/QUEUES    FILE/TTY
Thu Jun 29 11:58:57 2000   Interval:  2         Cswitch      36  Readch     3424
                                                Syscall     158  Writech    3622
Kernel    0.1   |                          |    Reads        22  Rawin         0
User     50.3   |##############            |    Writes        3  Ttyout      236
Wait      0.0   |                          |    Forks         0  Igets         0
Idle     49.5   |##############            |    Execs         0  Namei        33
                                                Runqueue    2.0  Dirblk        0
Interf   KBPS   I-Pack   O-Pack   KB-In  KB-Out Waitqueue   1.2
tr0       3.5      3.0      3.0     0.1     3.4
en0       0.0      0.0      0.0     0.0     0.0 PAGING           MEMORY
                                                Faults        0  Real,MB     511
Disk    Busy%     KBPS     TPS KB-Read KB-Writ  Steals        0  % Comp     18.0
hdisk0    0.0      0.0     0.0     0.0     0.0  PgspIn        0  % Noncomp  27.0
hdisk1    0.0      0.0     0.0     0.0     0.0  PgspOut       0  % Client    0.0
hdisk2    0.0      0.0     0.0     0.0     0.0  PageIn        0
hdisk3    0.0      0.0     0.0     0.0     0.0  PageOut       0  PAGING SPACE
                                                Sios          0  Size,MB    1024
tctestprg(13888)100.0% PgSp: 0.0mb root                          % Used      0.1
tctestprg(15752)100.0% PgSp: 0.0mb root                          % Free     99.8
snmpd    (7834)   1.0% PgSp: 0.9mb root
topas    (16022)  0.5% PgSp: 0.4mb root
gil      (2064)   0.0% PgSp: 0.3mb root            Press "h" for help screen.
topas    (13534)  0.0% PgSp: 0.2mb root            Press "q" to quit program.
```

Figure 20. topas command output

On the left side of command output there is a variable section. The first is a subsection that lists the CPU utilization in both numeric and block-graph format:

```
Kernel    5.0   |#                             |
User     49.7   |##############                |
Wait     16.1   |#####                         |
Idle     29.0   |########                      |
```

Following is the subsection with network interfaces statistics:

```
Interf   KBPS   I-Pack  O-Pack   KB-In  KB-Out
tr0       1.0      3.3     2.3     0.1     0.9
en0       0.0      0.1     0.0     0.0     0.0
```

Next, is physical disks statistics subsection. You can choose the maximum number of disks shown:

```
Disk    Busy%     KBPS     TPS KB-Read KB-Writ
hdisk0    0.0      0.0     0.0     0.0     0.0
hdisk1    0.0      0.0     0.0     0.0     0.0
hdisk2    0.0      0.0     0.0     0.0     0.0
hdisk3    0.0   4056.6    96.9     0.0  4056.6
```

The last subsection shows process information. Retrieval of process information constitutes the majority of the `topas` overhead:

```
tctestprg(15752)100.0% PgSp: 0.0mb root
tctestprg(13888)100.0% PgSp: 0.0mb root
topas    (16022)  0.5% PgSp: 0.4mb root
cp       (18112)  0.0% PgSp: 0.1mb root
cp       (15558)  0.0% PgSp: 0.1mb root
cp       (13662)  0.0% PgSp: 0.1mb root
```

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 Show all of the variable sections (network, disk, and process) if space allows.

d Show disk information. If the requested number of disks and the requested number of network interfaces will fit on a 25-line display, both are shown. If there is space left on a 25-line display to list at least three processes, as many processes as will fit are also displayed.

h Show the help screen.

n Show network interface information. If the requested number of disks and the requested number of network interfaces will fit on a 25-line display, both are shown. If there is space left on a 25-line display to list at least three processes, as many processes as will fit are also displayed.

p Show process information. If the requested number of processes leaves enough space on a 25-line display to also display the requested number of network interfaces, those are shown. If there is also space to show the requested number of disks, those are shown as well.

q Quit the program.

You can also set the command output using specified flags at the command line:

-i Sets the monitoring interval in seconds. The default is 2 seconds.

-n Specifies the maximum number of network interfaces shown. If a value of zero is specified, no network information will be displayed.

-p Specifies the maximum number of processes shown.If a value of zero is specified, no process information will be displayed.

6.16 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 file set.

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:

```
# /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 repage 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 25has the list of flags and some of the limitations for the settings.

Table 25. The vmtune command flags

Flag	Description
-b numfsbuf	Specifies the number of file system bufstructs. The current default in AIX Version 4 is 93 since 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 writes 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/Os 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 16KB 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 like 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.

Flag	Description
-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, 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.
-k npskill	Specifies the number of free paging-space pages at which AIX begins killing processes. The formula to determine default value of npskill in AIX Version 4 is: `MAX(64, number_of_paging_space_pages/128)` The npskill value has to 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 4KB pages) of the least recently used (lru) page replacement bucket size. This is the number of page frames which will be examined at one time for possible pageouts 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 much like a true lru algorithm. The default value is 512MB and the minimum is 256MB. Tuning this option is not recommended.
-M maxpin	Specifies the maximum percentage of real memory that can be pinned. The default value is 80%. If this value is changed, the new value should ensure that at least 4MB 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 few 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 2GB, then pd_npages is, by default, 524288.

Flag	Description
-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% of memory.
-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% 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% 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 through 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 through 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).

Flag	Description
-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 has to be greater than 0 and less than the total number of paging space pages on the system.The default value is 512.
-W maxrandwrt	Specifies a threshold (in 4KB pages) for random writes to accumulate in RAM before these pages are syncd to disk via 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.

6.17 Collecting data using the svmon command

The `svmon` command is used to display information regarding the current memory state. Although it is a complicated command it is useful command and the support professional needs to understand what it can do to assist in performance checking.

The `svmon` command generates seven types of reports:

- global
- user
- command
- workload management class
- process
- segment
- detailed segment

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

With the exception of the svmon -G and svmon -D reports the other command report options use the same flags with the same use of the flag. The difference for each command is in the output given by the command. In the following sections an example of the command and the out that is received from it will be explained. This will not be an exhaustive list of functions for each command, rather a list that will give some of the differences in out depending on the flag.

6.17.1 The svmon global report

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

The syntax for the `svmon -G` command is as follows:

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

Running the `svmon -G` global report command will give 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 giving the maximum memory allocated will have 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
```

where each line has the following meaning:

- 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` 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

6.17.2 The svmon user report

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

The command syntax for the `svmon -U` command is as follows:

```
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 has the following output:

```
# 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 column headings in a user login report are:

- 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 this columns 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 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 system that are basically 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 `-l` flag is specified, then for each segment in the last category, the list of process identifiers that use the segment is displayed. Beside the process identifier, the login user name that executes it is also displayed. See the `-l` flag description for special segments processing.

6.17.3 The svmon process report

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

The syntax for the `svmon -P` command is as follows:

```
svmon [-P [pid1...pidn] [-u|-p|-g|-v] [-ns] [-wfc] [-t Count] [ -i Intvl
[NumInt vl] ] [-l][-z][-m] ]
```

The `svmon -P` process report command has the following output:

```
# 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.

```
# 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 a meaning as described below:

- 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).

6.17.4 The svmon segment report

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

The `svmon -S` command syntax is as follows:

```
svmon [-S [sid1...sidn] [-u|-p|-g|-v] [-ns] [-wfc] [-t Count] [ -i Intvl
[NumInt vl] ] [-l] [-z][-m] ]
```

The `svmon -S` command has 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 5 working segments according to the number of virtual pages do the following:

```
# 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 type the following:

```
# 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
```

Its column headings are described as follows:

- 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 '-'. In that case, the exact esid values can be obtained through -P

option applied on each process identifiers 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 i-node number of the associated file are displayed, separated by a colon. (The device name and i-node 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 an other 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 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).

> **Note**
>
> VMM manages this value for statistics purpose. It may happened it is not updated. Then its value may be less than the inuse counters.

- Address Range Specifies the range(s) within the segment pages have been allocated. 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.

6.17.5 The svmon detailed segment report

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

The svmon -D syntax is as follows:

```
svmon [-D sid1...sidn [-b] [ -i Intvl [NumIntvl] ][-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 the command is as follows:

```
# 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 are detailed as follows:

The segid, type, address range, size of page space allocation, virtual and inuse headings are explained at the end of Chapter 6.17.4, “The svmon segment report” on page 191.

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

6.17.6 The svmon command report

The command report will give usage of specific commands being run. The command report is printed when the `-C` flag is specified.

The svmon -C command syntax is as follows:

```
svmon [-C cmd1...cmdn [-u|-p|-g|-v] [-ns] [-wfc] [-t Count] [ -i Intvl
[NumIntvl] ][-d][-l][-z][-m] ]
```

To check the memory usage of specific commands type the following:

```
# 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 it will give an error as shown above.

To check a command and display the memory statistics for the command type 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 in a command report are as follows:

- 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 columns 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 heading of the processes as described in 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 system that are basically shared by all processes.
  2. The segments that are only used by the set of processes
  3. 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. Beside the process identifier, the command name it runs is also displayed. See the `-l` flag description for special segments processing.

6.17.7 The svmon workload management class report

The workload class report is printed when the `-W` flag is specified.

The `svmon -W` command syntax is as follows:

```
svmon [-W [class1...classn] [-u|-p|-g|-v] [-ns] [-wfc] [-t Count] [ -i
Intvl [NumIntvl] ][-l][-z][-m] ]
```

Note

Before the svmon -W command can be run the Work Load Manager must be started.

The column headings in a workload class report are:

- 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 columns heading is displayed, svmon displays information about the segments belonging to the workload class.
  If -l option is specified, then for each segment, the list of process identifiers that use the segment is displayed. Beside the process identifier, the workload class the process belongs to is also displayed. See also -l flag description for special segments processing.

Note

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

6.17.8 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 26. The svmon command flags

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) are 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 identifier 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 basename 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 name 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 detail status of all 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.

Flag	Description
-w	Indicates that only working segments are to be included in the statistics. By default all segments are analyzed.
-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 are sorted in decreasing order by the total number of pages pinned.
-g	Indicates that the object to be printed are 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 are 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 at 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. **Note:** This flag should be used with caution because of its performance impacts.
-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 belong 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 system

Flag	Description
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 process.
-z	Displays the maximum memory size dynamically allocated (malloc) by svmon during its execution.
-m	Displays information about source segment rather than mapping segment when a segment is mapping a source segment.
-d	Displays for a given entity, the memory statistics of the processes belonging to the 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.

6.18 Quiz

6.18.1 Answers

6.19 Exercises

1. Name the seven types of reports the svmon command creates and a brief description of each.
2. What `svmon` command flags are used to display a programs resource utilization.
3. What `svmon` command flags are used to display a user resource utilization on the system.

6.20 Collecting data using 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.

Below is an example of the `vmstat` command without any flags:

```
# 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 number of seconds apart. If the Interval parameter is specified without the Count parameter, reports are continuously generated. A Count parameter of 0 is not allowed.

Below 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, which the `vmstat` command accesses through the use of the knlist subroutine and the /dev/kmem pseudo-device driver. The disk input/output statistics are maintained by device drivers. 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 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
```

The following example of a report generated by the `vmstat` command contains the column headings and their description:

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

- `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/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 64MB of memory or more, the minimum value (MINFREE) is 120 frames. For systems with less than 64MB, the value is two times the number of MB of real memory, minus 8. For example, a system with 32MB 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.

- `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.

• `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 wait and there is pending disk input/output.

• `Disk` 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 standard output the contents of the sum structure, which contains an absolute count of paging events since system initialization. The -s option is exclusive of the other `vmstat` command options.

Below 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
```

These events are described as follows:

• 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 page faults	Incremented if the page fault is to working storage and can be satisfied by assigning a frame and zero-filling it.
• executable-filled page faults	Incremented for each instruction page fault.
• pages examined by the 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. This count should equal the sum of page-ins and page-outs.
• 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 on each hardware interrupt.
• software interrupts	Incremented on 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 — Not maintained by the AIX operating system.
- syscalls — Incremented for each system call.

In example below an idle system will be shown and then load will be put onto the system and the resultant output will be analyzed to investigate potential problems.

Below 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 can be seen kthr (kernel thread) r (runable 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 zero.

In the memory column, the avm (average paging space memory) stayed relatively stable but the fre (free memory frames) value dropped from 1749 to it's lowest of 120. If the fre value had dropped below 120 for extended period of time this system would be continuously be 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 5, 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 an 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%.

6.21 Quiz

6.21.1 Answers

6.22 Exercises

The following are some exercises to ensure comprehension of this chapter.

1. When running the vmstat command there are either five or six measurement columns, describe all six and show the flag required for the sixth column.
2. Describe how a system could be described as CPU bound using the vmstat command and also what the percentages are for single and multi processor systems.

Chapter 7. Performance commands: iostat, lockstat

The following sub chapter are to be inserted into the global performance commands chapter.

7.1 The iostat command

The `iostat` command is a useful tool to give a first indication of I/O related performance problems. Iostat is capable of reporting CPU statistics, terminal I/O statistics, and disk I/O statistics, which can help identifying the I/O load on individual components such as a hard disks.

The 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:

```
iostat [-d|-t][physicalVolume ...][interval|count]
```

Table 27. Commonly used flags of the iostat comman

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 specifying the physical volume (PV) name of the individual disk or CD-ROM, iostat generates an I/O report only of these drives. If no PVs are specified the iostat 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.

Example:

```
# 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 which is updated every second (interval=1) and has only two reports (count=2).

Each report is combine 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 drive From the above example you can that the hdisk1 has been the most actively used drive.

The second report shown is the actual

During the report, there was a copy command started.

7.1.1 Historical disk I/O

In AIX Version 4.3 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.

> **Note**
>
> If the disk I/O history is disabled, iostat 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 which can be enabled or disabled in SMIT using: `smit chgsys`. The Figure 21 shows the corresponding SMIT screen.

```
                    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       [128]                    +#
  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           false                   +
  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          524288
  State of system keylock at boot time                normal
  Enable full CORE dump                               false                   +
  Use pre-430 style CORE dump                         false                   +
  CPU Guard                                           disable                 +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 21. SMIT screen for changing characteristics of operating system.

Remember when the historical disk I/O is activated to ignore the first report if you are looking for the real-time behavior of your system.

7.1.2 TTY and CPU utilization report

The first report section displayed by iostat contains the TTY and CPU utilization report.

The following columns are displayed:

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 and the system did not have an outstanding disk I/O request.

% iowait Shows the percentage of time that the CPU or CPUs were 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, like serial terminals, modems, FAX and so on. Pseudo TTY devices are telnet sessions and xterms (or other X based terminal emulators like dtterm and aixterm).

As the processing of characters 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 theses 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. More information about TTY performance, especially when your system has multiport adapters (8-, 16-, or 64-port adapters) can be found in *AIX Versions 3.2 and 4 Performance Tuning Guide*, SC23-2365, chapter *Tuning Asynchronous Connections for High-Speed Transfers*.

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.

When the iostat report shows that a CPU-bound situation does not exist with a high % idle, and % iowait time is greater than 25 percent, this might point to an I/O or disk-bound situation.

Example extracted from iostat 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
```

Following 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 like filemon or fileplace, see also Chapter 5, “LVM and JFS performance tools” on page 107.

Alternatively the iostat command can be used for determining that a performance problem is related to 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 CPU bound problem is when % iowait time is zero, when the system is not idle (% idle = 0).

To investigate if a system does not have a memory problem verify that physical volume that is used for swapping does not have an excessive load. Use the command lsps -a to determine the physical volume of the swap area.

7.1.3 iostat 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.

In previous releases of AIX, before 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 commands vmstat and iostat simply reported the CPU break down 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. 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 was busy in kernel mode at the time of the clock interrupt, the sys category gets the tick. If the CPU was 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 was, or 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 much more correct wio times when just a few threads are doing I/O and the system is otherwise idle.

7.1.4 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 flag -d. 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 followed by a number. Per 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 to a new location. An increase in disk active time percentage implies a performance decreases and response time increases. 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 written) 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 via 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 the using the different data columns just described it is important to notice the patterns and relationships between the data types.

A common relationship is between disk utilization %tm_act and data transfer rate tps. Generally you do not need to be concerned about a high disk busy rate %tm_act as long as the tps rate is also 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.

For example, if an application reads/writes sequentially, you should expect a high disk transfer rate (tps) when you have a high disk busy rate (%tm_act).

Kb_read and Kb_wrtn can confirm an understanding of an applications 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 on 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 older 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 Kbps 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. For some different SCSI types this is:

- SCSI-1 throughput rate of 3.5 MB/s (70% of 5 MB/s)
- SCSI-2 throughput rate of 7 MB/s (70% of 10 MB/s)
- Ultra SCSI throughput rate of 28 MB/s (70% of 40 MB/s)
- Ultra2 SCSI throughput rate of 56 MB/s (70% 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 refer to the *AIX Versions 3.2 and 4 Performance Tuning Guide*, SC23-2365, chapter *Monitoring and Tuning Disk I/O*.

Note

Like vmstat, iostat can only give a first indication about a performance bottleneck. The system administrator will have to use more indepth analysis tools like filemon to identify the source of the slowdown, see also Chapter 5, “LVM and JFS performance tools” on page 107.

7.2 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 lockstat can be used, you must create as root a new bosboot 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:

```
lockstat [ -a ][ -c LockCount ] [ -b BlockRatio ] [ -n CheckCount ] [ -p
LockRate ] [ -t MaxLocks ] [ Interval [Count ] ]
```

Table 28. Flags of the lockstat comman

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.
-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 which 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.

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.

If the lockstat command is executed with no options, an output similar to the following would be displayed.

Example:

```
# 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 report contains 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 cause the calling thread to sleep.

Some common subsystems are:

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:

```
# 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 gives 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 3-5 percent. Also be aware that AIX trace buffers will fill up much quicker when using this option since there are a lot of locks being used.

Chapter 8. CPU testcase

In this section a basic CPU bound performance problem scenario is shown, with conclusions made based on output from commands previously discussed in this book.

The environment consists of a 2-way F50 with 50 Netstation clients connected over Ethernet. Users are using a HTML application as 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 start up 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 really 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
```

Five jobs on the runqueue is not a normal state for this system. Next task 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   lft0 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 testprograms uses abnormally much CPU.

There are several ways to go (`kill PID`, for example), but the proper thing would be to check with the user thomasc - what are these process; why are they running - can they be stopped; do they have to run now - can they be rescheduled. Rescheduling this kind of CPU consuming process 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`. The thing to remember is to start such jobs at times when they are not in conflict with OLTPs.

If they have to run now, and to be running at times in the future, some changes has to be done in the system. The best thing would probably be moving these test programs to a test system, excluded from the production environment.

Another way to go is to buy more CPUs, which is a nice thing to do, but may move the bottleneck to another resource of the system, for example memory.

Finally, implementing the workload manager (WLM) introduced with AIX 4.3.3, may help the problem by moving placing these test programs in a lesser prioritized group. For more information on WLM see *AIX Workload Management*, SG24-5977-00.

Chapter 9. Examining I/O performance problem scenarios

9.1 I/O Bottlenecks

The following scenarios are examples of various command reports used as input for determining system which have an I/O performance bottleneck.

9.1.1 Scenario1

The following scenario provides the following `vmstat` 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 take over a period of 20 min using a 120 second interval. The first figures which are interesting in this reports 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% and peaking at 92% on the second last measurement. The average I/O wait time over the sample period is 72%, 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.

Notice also that the b value in the kernel thread column is also quite high. The b column lists the average number of kernel threads placed on the wait queue per second. 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 all are zero and the list of free memory pages (fre) is still acceptable.

To determine more information on where the possible I/O bottleneck an `iostat` command should be run on this system. This will provide information about on how the disk I/O is distributed between the physical volumes and which where the possible bottlenecks.

9.1.2 Scenario2

This scenario provides the following lsps and iostat report as input:

```
# 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 minutes period in 120 seconds interval. The first report is not shown and should be ignored for real-time analysis as it is the historical disk I/O. For details on the iostat command report see also Chapter 7.1, “The iostat command” on page 215.

Notice the high I/O wait time (% iowait), which is increasing from 5.1% to 30%. Generally if an I/O wait time exceeds 25% there is an problem related to disk I/O. There might be cases where I/O wait time is 0% 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% and 99.8%, which is almost constantly active. This indicates that the I/O bottleneck is bound to the disk hdisk0. Notice the the Kb_wrtn value indicating the data written to the disk is fairly high compared amount of data read from the disk. This leads to the conclusion that the I/O limits of the hdisk0 are reached. To improve the I/O performance the other disks should be used more actively, by either moving hot file, filesystem and logical volumes to the less active disks. In general a more insight investigation is required using other tools like `filemon` and `lslv`.

9.2 Logical volume performance problem scenarios

The following scenarios are examples of various command reports used as input for determining performance problems related to logical volumes.

9.2.1 Logical Volume Fragmentation scenario

The `lslv` command is used for displaying the attributes of logical volumes along with other information, such as the fragmentation of a logical volume on a physical volume.

Following list 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 lv00 in the above example consisting of 71 logical partitions LPs is fragmented over four of the file possible intra disk allocation sections. The sections outer edge and outer middle are allocated from LP 01-32 and 33-64 respectively on similar contiguously physical partitions. The LP 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 decrease in I/O performance, due to longer seek and disk head changes for sequential read/write operation in the last part of the logical volume. As there is free space left on the physical volume to reorganizing the lv00. Use the `reorgvg` command on this logical volume would help improving the performance of lv00.

For more information on the `lslv` command refer to Chapter 5.2, “LVM performance analysis using lslv” on page 108.

9.2.2 Monitoring scenario using filemon

Consider a system with the following disks available:

```
# lspv
hdisk0          000bc6fdc3dc07a7    rootvg
hdisk1          000bc6fdbff75ee2    none
```

The following output shows an extraction of a filemon report made for monitoring the logical volumes of a system:

```
...
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 report was generated with the filemon command: filemon -O lv -o filemon.out.

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, it would be possible to add this physical volume to the rootvg an 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.

9.2.3 Logical volume allocation problem scenario

The following scenario shows a series of status commands from a system with an allocation problem on an 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 on the hdisk0 of the rootvg.

Following volume group information is present:

```
# 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.

The physical volumes hdisk0 contains:

```
# 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 disks contain:

```
# 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 show:

```
# 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 the each of the LVs on a corresponding physical volume. Alternatively the Inter disk allocation policy could have been set to maximum and limiting 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 respectively.

A modification to this can also be done if the LV is already created 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 filesystem could be created. By defining a dedicated jfslog for both /db02 and /db03 would improve the performance. In this way the different filesystem residing on their individual disks would not utillize the hdisk1 for the jfs logging, but rather their own physical volume.

Chapter 10. I/O performance problem scenario two

In this scenario a user complains that when the month end report is being run, the 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.

10.1 The system output

In this section are the outputs of the system gathered by the `vmstat` and `iostat` commands.

Example:

Below is the output of the system using the `iostat` command.

```
# 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
```

10.2 The output investigation

In this section the key indicators of the output will be looked at and an explanation given regarding the output.

10.2.1 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%` (add the column excluding the first line and divide by nine). If the `wa` output is higher than 40% it may indicate a problem with the disk subsystem.

10.2.2 The iostat command output investigation

The key values to check here are the `% iowait` and the `% tm_act` values.

10.2.2.1 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%` (add the column excluding the first line and divide by nine). If the systems `% iowait` is higher than 25% it is advisable to investigate problem and take corrective action.

10.2.2.2 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% for hdisk0 and and average of 99.7% for hdisk1. If the percentage for the time a disk is busy is high on a smaller system with less disks there will be noticeable performance degradation on the system. On average a system that is performing at 40% average per disk is running with good throughput. This is however not always possible with smaller systems with less disks.

10.3 Recommendations

Below 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 idles 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 which can be run earlier or later, this way the load is also spread across time.

Chapter 11. Paging performance problem scenario

In this chapter paging performance will be investigated. The symptoms of high memory utilization will be described and possible corrective action that could be taken will be explained.

11.1 The system output

In this section are the system outputs gathered by the `svmon` and `vmstat` commands.

Example:

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 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 3598 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 1385 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.

Output of the `ps` command:

```
# 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
```

Output of the `svmon` command:

```
# svmon -i 5 3
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
```

Output showing the top ten memory using processes using the `svmon` command:

```
# 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 `lspas -a` 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.

11.2 The output investigation

From the above output an investigation can be done on the various components within the system to determine the areas that are causing performance problems. For the investigation output that will be used will mostly be from the `vmstat` command output. The other commands output is for information and confirmation.

11.2.1 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.

11.2.2 The memory column

The `memory` column displays information about the use of real and virtual memory. A page size 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 `lsps -a` command. When all the paging space has been utilized or reaches 100%, 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.

11.2.3 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 necassarily 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 reflected 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 that 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.

11.2.4 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.

11.2.5 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%.

The `sy` column indicates the amount of time spent in system mode. In this example it is never below 92%.

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 7%.

Note

In a single processor system if `us+sy` is greater than 90% the system can be considered CPU bound.

11.3 Recommendations

In the example there are some recommendations that can be implemented to ease the situation.

- If it is possible an additional CPU can be added to try and get the cpu utilization below 80%.
- 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 only occurs during a certain period of time it is advisable to set the system to perform large functions that utilize large amount of resources and spread them out not to conflict with one another.

Appendix A. Error log

The following topics are discussed in this chapter:

- General discussing about error logging subsystem
- Managing error log
- Reading error logs

The error log is the first place which an administrator will search for cause of improper system work.

A.1 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 time stamp 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
```

A.2 Managing 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 `/usr/lib/errdemon` command starts error logging and initialize /var/adm/ras/errlog as a default log file.

A.2.1 Configuring Error Logging

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 `/usr/lib/errdemon -l`. 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 display 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 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 you can check what you did:

```
# /usr/lib/errdemon -l
Error Log Attributes
--------------------------------------------
Log File                /var/adm/ras/errlog.new
Log Size                2000000 bytes
Memory Buffer Size      16384 bytes
```

A.2.2 Clearing 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 errors entries use the following command:

```
# errclear -d S 0
```

Alternatively, use the `smitty errclear` command.

A.3 Reading error logs in details

You can generate an error report from entries in an error log. There are two main ways of viewing the error log:

- The easiest way to read error log entries is `smitty errpt` command. Output from this command is show in the Figure 22 on page 257.
- The second way to display error log entries is `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. 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
```

```
                         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 22. smitty errpt output

A.3.1 The errpt command output

Report obtained from `errpt` command without any flags contain the following informations.

A.3.1.1 Identifier

Numerical identifier for the event.

A.3.1.2 Timestamp

Time when the error occur in following format mmddhhmmyy. The following timestamp `0627172400` means that error occur June, 27th at 17:24 (5.24 PM) year 00 (year 2000).

A.3.1.3 Type

Severity of the error that has occurred. Five types of errors are possible:

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 Condition 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 Condition 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.

A.3.1.4 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).

A.3.1.5 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 to be used to analyze the error.

A.3.1.6 Description

Brief summary of the error.

A.3.2 Examples of formatted output from errpt command

- To list all of 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 have 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:      006151424C00
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 the June, 26th 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
```

A.4 Commands

For a complete reference of the following command use the *AIX Version 4.3 Command Reference* or the online man pages.

A.4.1 errpt

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 ]
```

Table 29. 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.
-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.

A.5 References

The following publications contain more information about system error logging.

- *AIX Version 4.3 Problem Solving Guide and Reference*, SC23-4123.
- *AIX Version 4.3 Commands Reference, Volume 2*, SC23-4116.
- *AIX Version 4.3 Files Reference,* SC23-4168.

A.6 Quiz

A.6.1 Answers

A.7 Exercises

1. Change default error log attributes.
2. Using `errpt` command, generate a report of errors caused by errdemon resource.
3. Using `errpt` command generate a report of software errors but limit it to temporary errors.
4. Generate the same reports using `smitty` tool.
5. Delete all software logs.

Appendix B. Using the additional material

This redbook also contains additional material in CD-ROM or diskette format, and/or Web material. See the appropriate section below for instructions on using or downloading each type of material.

B.1 Using the CD-ROM or diskette

The CD-ROM or diskette that accompanies this redbook contains the following:

File name	*Description*
????????.cpp	????Code Samples????
????????.html	????HTML Documents????
????????.prz	????Presentations????

B.1.1 System requirements for using the CD-ROM or diskette

The following system configuration is recommended for optimal use of the CD-ROM or diskette.

Hard disk space:	????MB minimum????
Operating System:	????Windows/UNIX/S390????
Processor:	???? or higher????
Memory:	????MB????
Other:	????CD-ROM drive????

B.1.2 How to use the CD-ROM or diskette

You can access the contents of the CD-ROM or diskette by pointing your Web browser at the file ????index.html???? in the CD-ROM or diskette root directory and following the links found there. Alternatively, you can create a subdirectory (folder) on your workstation and copy the contents of the CD-ROM or diskette into this folder.

B.2 Locating the additional material on the Internet

The CD-ROM, diskette, or Web material associated with this redbook is also available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG24????`

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number.

B.3 Using the Web material

The additional Web material that accompanies this redbook includes the following:

File name	*Description*
????????.zip	????Zipped Code Samples????
????????.zip	????Zipped HTML Documents????
????????.zip	????Zipped Presentations????

B.3.1 System requirements for downloading the Web material

The following system configuration is recommended for downloading the additional Web material.

Hard disk space:	????MB minimum????
Operating System:	????Windows/UNIX/S390????
Processor:	???? or higher????
Memory:	????MB????

B.3.2 How to use the Web material

Create a subdirectory (folder) on your workstation and copy the contents of the Web material into this folder.

Appendix C. Special notices

This publication is intended to help ???who?????? to ??do what????? The information in this publication is not intended as the specification of any programming interfaces that are provided by ?????????product name(s)????????????. See the PUBLICATIONS section of the IBM Programming Announcement for ???????product name(s)???????????? for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers

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.

Appendix D. Related publications

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

D.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 271.

- *RS/6000 Performance Tools in Focus*, SG24-4989-00
- *AIX Logical Volume Manager, from A to Z: Introduction and Concepts*, SG24-5432
- *????full title???????*, xxxx-xxxx

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at **ibm.com**/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *????full title???????*, xxxx-xxxx
- *????full title???????*, xxxx-xxxx

- *????full title???????*, xxxx-xxxx

D.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://????????.???.???/` description
- `http://????????.???.???/` description
- `http://????????.???.???/` description

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** **ibm.com**/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Abbreviations and acronyms

IBM	International Business Machines Corporation
ITSO	International Technical Support Organization

Index

Symbols

/etc/hosts 151
/etc/netsvc.conf 151
/etc/rc.nfs 153
/etc/resolve.conf 151
__prof.all 103

A

adapter
    SCSI bottleneck 221
allocation
    logical volume 113
allocation policy
    intra disk 110
application performance 126
at 229
attributes
    logical volume 108
availability
    logical volume manager 110

B

bad block policy 109
base value 79
batch 229
BB policy 109
bigfile file system 125
bosboot 222
bottleneck
    SCSI adapter 221

C

center disk allocation 110
chdev 138, 139, 150
chnfs 152, 154
collecting
    disk I/O history 217
collecting data
    sar command 159
    svmon 183
commads
    netpmon 73
    nice 90
commands 139
/etc/netsvc.conf 151
/etc/resolve.conf 151
at 229
batch 229
chdev 138, 139, 150
chnfs 152, 154
defragfs 128
emstat 101
entstat 138
filemon 69, 115, 131
fileplace 69, 126, 131
host 151
ifconfig 138, 139, 150
installp 40, 41
iostat 68, 215, 227
iptrace 146
lockstat 222
lsattr 138, 150
lsdev 249
lslpp 39, 42, 101
lslv 69, 108, 132
lsps 62, 249
migratepv 238
netpmon 145
netstat 71, 138, 142, 173
nfso 139
nfsstat 72
nice 54, 87
no 136, 143
nslookup 151
oslevel 39
ping 142
ps 53, 62, 78, 93, 228, 244
renice 54, 87, 89, 90
reorgvg 234
rmss 63
sa1 170
sa2 170
sadc 170
sar 51, 159, 163, 173
schedtune 54, 84
svmon 62, 183, 186, 189, 191, 194, 196, 245
tcpdump 139, 146
time 52, 227
tokstat 138
topas 173
    command output 173

tprof 53, 103
trcstop 145
vmstat 53, 60, 101, 141, 173, 205, 228, 243
vmtune 63, 67, 177, 211, 249
computational memory 59
copies
logical volume 109
CPU
bound problem 218
iostat utilization report 217
statistics with iostat 218
CPU bound 49
process and thread table 51
process state figure 50
processes 49
threads 49
CPU penalty 79
CPU testcase 227
at 229
batch 229
iostat 227
ps 228
recent CPU usage 229
rescheduling 229
time 227
vmstat 228
WLM 229
crontab 229

D
DEFAULT_NICE 79
deferred page space allocation 60
defragfs command 128
de-fragmentation of file system 128
detailed file stats report in filemon 119
disk
I/O statistics with iostat 215
I/O wait 142
iostat utilization report 220
unbalanced load 219
disk bound 63
logical volume device driver 63
logical volume manager 63
lvdd figure 64
disk bound problem 218
disk bound problems 123
distribution column in lslv -l 113
DPSA 60

E
early alloaction algorithm 60
emstat 101
emulation routines 101
entstat 138

F
figures
128 run queues 78
code,data.private and shared segments 57
CPU penalty 86
Disk, LVM and file system levels 107
global run queue 77
JFS file system organization 125
lvdd 64
LVM figure 65
LVM intra disk positions 111
memory registers 58
muliple run queues 82
network parameters 71
performance tuning flowchart 49
process state 50
VMM segments 55
file memory 59
file system
bigfile file system 125
de-fragmentation 128
fileplace command 126
fragmentation size 124
i-node 124
journaled file system - JFS 124
logical fragment 127
organization 124
performance 124
recommendations 130
filemon 69
command 115, 131
detailed file stats report 119
disk access 123
frequently accessed files 123
logical file system 115
logical volume monitoring 116
monitoring scenario 235
most active files report 119
physical volume monitoring 116
report
logical file level 117
logical volume level report 120

physical volume level 121
virtual memory level 122
report analysis 117
virtual memory system monitoring 116
fileplace 69
fileplace command 126, 131
files
/etc/hosts 151
/etc/rc.nfs 153
/usr/include/sys/lockname.h 224
__prof.all 103
fragmentation
fileplace command 126
logical volume fragmentation scenario 234
fragmentation of logical volume 112
fragmentation size 124
fragmented files 126
free list 58
frequent periodic load balancing 83
frequently accessed files 123

G
global run queue 77

H
hash anchor table 59
high-water mark 67
historical disk I/O 216
host 151

I
I/O bottlenecks
scenarios 231
I/O pacing 67
I/O performance
problem scenarios 231
idle load balancing 83
ifconfig 138, 139, 150
in band column in lslv -l 112
infrequent periodic load balancing 84
initial load balancing 83
inner edge disk allocation 110
inner middle disk allocation 110
i-node 124
installp 40, 41
inter disk policy 111
for logical volume 109
maximum 111
minimum 111
intra disk
allocation 110
policy
center 110
inner edge 110
inner middle 110
outer edge 110
outer middle 110
policy for logical volume 109
iostat 68, 173, 227
enhancement to 4.3.3 68
iostat command 215
historical disk I/O 216
SMP behaviour 219
IP 136
data flow 137
input queue 140, 144
ipqmaxlen 140, 144
iptrace 139, 146

J
JFS 65
bigfile file system 125
file system organization 124
fileplace command 126
fragmentation 65
fragmentation size 124
i-node 124
performance tools 107

K
kernel locks
display with lockstat 222

L
late allocation algorithm 59
limitations
striping 129
load balancing 83
frequent periodic load balancing 83
idle load balancing 83
infrequent periodic load balancing 84
initial load balancing 83
lock
display of lock contention with lockstat 222

lock contention
  display with lockstat 222
lockstat command 222
logical fragment 127
logical partition 64
logical volume 64
  allocation 113
  allocation scenario 235
  attributes 108
  bad block policy 109
  copies 109
  distribution 113
  fragmentation 112
  fragmentation scenario 234
  highest performance 115
  inter disk policy 109, 111
  intra disk policy 109
  mirror write consistency 109
  organization for highest performance 128
  relocateable 109
  scheduling policy 109
  stripe size 112
  stripe width 112
  striping 112, 129
  upper bound 109
  write policy
    parallel 110
    sequential 110
  write verify 109
logical volume device dirver 63
logical volume manager
  availability 110
  monitoring 115
  performance
    analysis with lslv 108
    tools 107
logical volume manger 63
low-water mark 67
lsattr 138, 150
lsdev command 249
lslpp 39, 42, 101
lslv 69
lslv command 108, 132
lsps 62
lsps command 249
LVDD 63
LVM 63
  dependecies figure 65
  fragmentation 65
  high-water mark 67
  I/O pacing 67
  JFS 65
  logical partition 64
  logical volume 64
  low-water mark 67
  maxpgahead 66
  minpgahead 66
  physical partition 64
  physical volume 64
  sequential-access read ahead 66
  volume group 64
  write-behind 67

M
maximum transfer unit (MTU) 135
maxpgahead 66
mbufs 139, 144
memory bound 54
  virtual memory 55
migratepv command 238
minpgahead 66
mirror write consistency 109
Mirroring 109
monitoring
  filemon command 115
  logical volume 115
  scenario with filemon 235
most active files report 119
MTU 150
multiple run queues 81
multiprocessor
  behaviour of iostat 219

N
name resolution
  performance 151
netpmon 73, 145
netstat 71, 138, 142, 173
network
  I/O wait 142
  tuning tools 149
network bound 69
  parameter figure 71
NFS 145
  client performance 153
  file system 154
  mount options 155

server performance 151
tuning 151
nfs_socketsize 153
nfs_tcp_socketsize 153
nfso 139
nfsstat 72
NICE 79
nice 54, 79, 87, 90
changing value on running thread 89
flag table 89, 90
running program with nice 87
no 136, 143
nslookup 151
NSORDER 151

O

organization
file system 124
oslevel 39
outer edge disk allocation 110
outer middle disk allocation 110
overhead
of performing tools 130

P

packet
dropped 139, 144
page fault 58
page frame table 58
page stealing 58
paging performance problem 243
investigation 249
recommendations 252
paging space
disk performance 130
parallel write policy 110
performace
filemon 115
performance
analysis and control 37
controlling resource allocation 49
CPU bound 49
define and prioritizie 48
disk bound 63
file system 124
file system recommendations 130
highest logical volume performance 115
identify resources 48
identify workload 47
load monitoring 37
logical volume manager
analysis with lslv 108
lvdd 64
LVM and JFS performance tools 107
LVM dependencies figure 65
memory bound 54
memory register figure 58
minimize requirements 48
network bound 69
network parameter figure 71
process and thread table 51
process state figure 50
processes 49
resource table 48
threads 49
tuning flowchart picture 49
VMM segment figure 55
VMM segment picture 57
vmstat table 61
performance capacity planning 37
performance tools 37
fileset
perfagent.server 38
perfagent.tool 38
perfagent.tools 39
perfmgr.common 38
perfmgr.local 37
perfmgr.network 38
installing 37
overview 37
Performance Toolbox 37
releases 38
Performance Toolbox Agent 37
Performance Toolbox Manager 37, 41
performance tuning definition 47
physical partition 64
physical volume 64
filemon report 121
unbalanced load 219
physical volume utilization 221
ping 142
priority 79
priority calculation 4.3.2 76
priority calculation 4.3.3 79
problems
disk bound 123
processes 49

processor
  601 PowerPC 101
  604 PowerPC 101
  POWER 101
    instructions 101
  PowerPC 101
prof
  total collumn 104
protocol
  statistics 142
protocols
  IP 136
  TCP 135
  tuning 139
  UDP 135
ps 53, 62, 78, 93, 228
  %CPU column 94
  %MEM column 96
  C column 93
  PGIN column 97
  RSS column 96
  SIZE column 95
  TIME column 94
  TRS column 97
  TSIZ column 97
ps command 244
PSALLOC 59
  Deferred Page Space Allocation 60
  early allocation algorithm 60
  late allocation algorithm 59

Q
queue
  receive 137
  size 138
  transmit 137

R
receive queue 136
recent CPU usage 78
registers 57
relocatable
  attribute for logical volume 109
renice 54, 87, 89, 90
  flag table 89, 91
reorgvg command 234
report
  iostat 216
    CPU utilization 217
    disk utilization 220
    TTY utilization 217
rescheduling 229
resent CPU usage 229
rfc1323 140
rmss 63
rx_que_size 136

S
sa1 command 170
sa2 command 170
sadc command 170
sar 51, 173
  /usr/lib/sa/sa1 51
  usr/lib/sa/sa2 51
sar command 159, 163
  flags 164
saturated SCSI adapter 222
sb_max 140, 141
scenarios
  filemon monitoring 235
  I/O bottlenecks 231
  I/O performance 231
  iostat 232
  logical volume allocation 235
  logical volume fragmentation 234
SCHED_D 79, 84
SCHED_FIFO 75
SCHED_FIFO2 76
SCHED_FIFO3 76
SCHED_OTHER 76
SCHED_R 79, 84
SCHED_RR 76
schedtune 54, 84
  commands
    schedtune 90
  example 1 84
  example 2 85
  example 3 85
  flag table 86
    tables
      schedtune flags 90
  reset 86
  SCHED_D 84
  SCHED_R 84
  SCHED_R and SCHED_D guidelines 86
scheduler

128 run queue figure 78
base value 79
CPU penalty 79
CPU penalty figure 86
DEFAULT_NICE 79
global run queue 77
global run queue figure 77
load balancing 83
multiple run queue figure 82
multiple run queues 81
NICE 79
nice value 79
priority 79
priority calculation 4.3.2 76
priority calculation 4.3.3 79
recent CPU usage 78
SCHED_D 79, 84
SCHED_FIFO 75
SCHED_FIFO2 76
SCHED_FIFO3 76
SCHED_OTHER 76
SCHED_R 79, 84
SCHED_R and SCHED_D guidelines 86
SCHED_RR 76
schedtune example 1 84
schedtune example 2 85
schedtune example 3 85
steal threshold 83
steal_max 83
waitproc 83
xnice 80
scheduling policy for logial volume 109
SCSI
adapter bottleneck 221
segment registers 57
sequential write policy 110
sequential-access read ahead 66
slowest average seek 111
SMIT fast path
smit chgsys 217
smitty
install_all 41
list_software 40
SMP
iostat behaviour 219
socket
active 142
buffer 151
buffer overflows 153
receive buffer 136, 140
send buffe 140
send buffer 135
statistics
CPU 145
queues 173
utilization 174
disk I/O 215
file 173
internet socket calls 145
mbufs 144
memory 173
network device-driver I/O 145
network interfaces 174
paging 173
physical disks 174
terminal I/O 215
statisticsprotocols 142
statistisc
ethernet 138
steal threshold 83
steal_max 83
stripe size logical volume attribute 112
stripe width logical volume attribute 112
striping 112
limitations 129
logical volume striping 129
recommendations 129
svmon 62
svmon command 183, 186, 189, 191, 194, 196, 245
command report 196
output description 198
syntax 196
detailed segment report 194
output description 196
syntax 194
flags 200
global report 183
output description 185
syntax 183
process report 189
output description 191
syntax 189
report types 183
segment report 191
output description 192
syntax 191
user report 186

output description 188
syntax 186
workload class report 199
output description 199
syntax 199
system
typical AIX system behaviour 123

T

tables
hardware and logical resources 48
nice flags 89, 90
processes and threads 51
renice flags 89, 91
schedtune flags 86
VMM related output from vmstat 61
TCP 135, 151
data flow 137
tcp_recvspace 140
tcp_sendspace 135, 140
tcpdump 139, 146
telnet
session 147
terminal
I/O statistics with iostat 215
I/O wait 142
testcase
CPU 227
thewall 140, 144, 150
threads 49
R state 50
ready to run 50
S state 50
sleeping 50
state 50
suspended 51
T state 51
throughput
SCSI adapters 221
time 52, 227
tokstat 138
tools
LVM and JFS performance tools 107
topas 173
command output 173
tprof 53, 103
FREQ collumn 105
general report 103
using tprof on a program 105
translation lookaside buffer 58
transmit queue 136
trcstop 145
TTY
devices 218
iostat utilization report 217
tx_que_size 136, 139

U

UDP 135, 151
data flow 137
udp_recvspace 140
udp_sendspace 135, 140
unbalanced disk load 219
upper bound attribute for logical volume 109
utilization
CPU with iostat 217
disk with iostat 220
TTY with iostat 217

V

virtual memory
client segment 56
code segment 56
computational memory 59
data segment 56
file memory 59
filemon report 122
free list 58
HAT 59
high-water mark 67
I/O pacing 67
low-water mark 67
maxpgahead 66
memory register figure 58
minpgahead 66
page fault 58
page stealing 58
persistent segment 55
private segment 56
PSALLOC 59
PTF 58
segment figure 55
segment picture 57
segment registers 57
segments 55
sequential-access read ahead 66

shared segment 56
TLB 58
vmstat table 61
working segment 55
write-behind 67
Virtual Memory Manager (VMM) 177
virtual memory monitoring
filemon command 115
vmstat 53, 60, 101, 141, 228
vmstat command 205, 243
cpu column description 251
faults column description 251
kthr column description 250
memory column description 250
output description 206
output interpretation 212
page column description 250
sum structure 208
vmstatcommands
vmstat 173
vmtune 63, 67
vmtune command 177, 211, 249
flags 178
syntax 177
volume group 64

W
waitproc 83
wirte-behind 67
WLM 229
workload
identify 47
write policy
parallel 110
sequential 110
write verify for logical volume 109

X
xnice 80

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6184-00 IBM Certification Study Guide AIX Version 4.3 Performance and System Tuning
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM

Redbooks

IBM Certification Study Guide AIX Version 4.3 Performance and

IBM

Redbooks

IBM Certification Study Guide AIX Version 4.3 Performance

IBM

Redbooks

IBM Certification Study Guide
AIX Version 4.3 Performance

IBM

Redbooks

IBM Certification Study Guide
AIX Version 4.3
Performance and System

(0.1”spine)	(0.2”spine)	(0.5” spine)	(1.0” spine)	(1.5” spine)
0.1”<->0.169”	0.17”<->0.473”	0.475”<->0.875”	0.875”<->1.498”	1.5”<-> 1.998”
53<->89 pages	90<->249 pages	250 <-> 459 pages	460 <-> 788 pages	789 <->1051 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set

IBM Certification Study Guide AIX Version 4.3

IBM Certification Study Guide AIX Version 4.3

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set

Draft Document for Review July 21, 2000 9:30 am

IBM Certification Study Guide AIX Version 4.3

Redbooks

The AIX & RS/6000 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 RS/6000. A complete set of professional certifications are available. They include:

IBM Certified AIX User
IBM Certified Specialist - RS/6000 Solution Sales
IBM Certified Specialist - AIX V4.3 System Administration
IBM Certified Specialist - AIX V4.3 System Support
IBM Certified Specialist - RS/6000 SP
IBM Certified Specialist - AIX HACMP
IBM Certified Specialist - Domino for RS/6000
IBM Certified Specialist - Web Server for RS/6000
IBM Certified Specialist - Business Intelligence for RS/6000
IBM Certified Advanced Technical Expert - RS/6000 AIX

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. These job incumbents bring a wealth of experience into the development process. Thus making the exams much more

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by IBM's 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

SG24-6184-00 ISBN