

COMPAQ

.....

Performance Tuning for SCO UNIX on Compaq Systems

Compaq TechNote

.....

NOTICE

The information in this publication is subject to change without notice.

COMPAQ COMPUTER CORPORATION SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL.

This publication contains information protected by copyright. No part of this publication may be photocopied or reproduced in any form without prior written consent from Compaq Computer Corporation.

The software described in this guide is furnished under a license agreement or non disclosure agreement. The software may be used or copied only in accordance with the terms of the agreement.

This publication does not constitute an endorsement of the product or products that were tested. The configuration or configurations tested or described may or may not be the only available solution. This test is not a determination of product quality or correctness, nor does it ensure compliance with any federal, state, or local requirements. Compaq does not warrant products other than its own strictly as stated in Compaq product warranties.

Product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

© 1995 Compaq Computer Corporation.
All rights reserved. Printed in the U.S.A.

Compaq, Fastart, Compaq Insight Manager, Systempro, Systempro/LT, SmartStart, and NetFlex
Registered United States Patent and Trademark Office.

ProLiant, ProSignia, Qvision, and Systempro/XL are trademarks of Compaq Computer Corporation.

Microsoft is a registered trademark of Microsoft Corporation and Windows and
Windows NT are trademarks of Microsoft Corporation.

Performance Tuning for SCO UNIX on Compaq Systems

First Edition (May 1993)
Part Number 145730-001

Contents

Chapter 1

Introduction

About This TechNote	1-1
Performance Analysis Tools	1-2
Capturing the Process State	1-3
Capturing System Activities and Statistics.....	1-4
Listing the Kernel Parameters.....	1-5

Chapter 2

Tuning the Memory Subsystem

Causes of Memory Subsystem Performance Bottlenecks	2-1
Not Enough Available Free Memory	2-1
Critical Memory Shortage and System Performance.....	2-2
Identifying Memory Subsystem Performance Bottlenecks	2-3
Determining Memory Allocation.....	2-3
Monitoring User Processes Virtual Memory	2-4
Monitoring Swap and Paging Activities.....	2-5
Finding the Amount of Free Memory	2-6
Investigating System Swap Activities.....	2-7
Investigating System Paging Activities.....	2-8
Guidelines for Tuning the Memory Subsystem.....	2-16
Adding Memory	2-16
Balancing the System Workload.....	2-21
Tuning the Kernel Parameters	2-21

Chapter 3

Tuning the Disk Subsystem

UNIX Filesystem I/O	3-1
Raw Device I/O.....	3-2
COMPAQ SMART SCSI Array Controller.....	3-3
Identifying Disk Subsystem Performance Bottlenecks	3-4
Block I/O	3-4
Swap and Paging	3-4
Raw Disk I/O	3-5
Monitoring Percentage Waiting for I/O	3-5
Monitoring Buffer Block I/O and Raw I/O	3-9
Monitoring Buffer Cache and Block I/O Activities	3-9
Monitoring Block I/O System Call Activities	3-11
Guidelines for Tuning the Disk Subsystem.....	3-15
Disk Access Patterns and Fault Tolerance Selection	3-16
Adding an Additional SMART Controller	3-19
Creating Logical Drives	3-20
Limiting Disk Fault Tolerance.....	3-22
Tuning the Kernel Buffer Cache Parameters.....	3-22
Maintaining Efficient Filesystems	3-24

Chapter 4

Tuning the System Processor

System Processor Activities	4-1
Identifying System Processor Performance Bottlenecks.....	4-2
Monitoring the System Load Average	4-3
Monitoring the Percentage of System Processor Use	4-3
Monitoring the System Workload.....	4-9
Guidelines for Tuning the System Processor.....	4-10
Adding a Second System Processor Board.....	4-10
Balancing the Workload Mix.....	4-10
Improving the Efficiency of User-Defined Features.....	4-11
Tuning the Kernel Parameters	4-12

Chapter 5

Performance Analysis Tools Reference

The PS Command	5-1
The SAR Command	5-4
Data Gathering	5-11
Exit Values.....	5-12
Diagnostic Messages	5-12
Command Examples.....	5-13
Warning	5-13
Files	5-13
Standards Conformance.....	5-13
The MPSAR Command.....	5-14
Exit Values.....	5-20
Diagnostic Messages	5-20
Command Examples.....	5-21
Warning	5-21
Files	5-21
Standards Conformance.....	5-21
The cpqmon Command	5-22
Basic Operation.....	5-22
Commands	5-23
Screen Commands.....	5-24
Command Line Options.....	5-27
Screen Information	5-28
Notes.....	5-34
Authors	5-34
The SYSDEF Command	5-35
The UPTIME Command	5-35

Chapter 1

Introduction

This COMPAQ TechNote is for system integrators or administrators with experience in configuring a system using SCO UNIX and a COMPAQ EISA-based product.

About This TechNote

This TechNote primarily addresses the operating system and hardware issues of performance management. Discussions on hardware and software architectures and configuration options exist only in relation to performance. Other factors (such as application design and software implementation techniques) that can contribute significantly to performance are beyond the scope of this TechNote.

Use the following table to locate the information you want in the remaining chapters:

Table 1-1
TechNote Chapter Summary

Chapter 2 TUNING THE MEMORY SUBSYSTEM	Describes the causes of performance bottlenecks associated with system memory and provides information about diagnosing and tuning.
Chapter 3 TUNING THE DISK SUBSYSTEM	Describes the causes of performance bottlenecks associated with disk subsystems and provides information about diagnosing and tuning.
Chapter 4 TUNING THE SYSTEM PROCESSOR	Describes the causes of performance bottlenecks associated with the system processor and provides information about diagnosing and tuning.
Chapter 5 PERFORMANCE ANALYSIS TOOLS REFERENCE	Provides reference information on the performance tools used in the TechNote.

You can also use the information in this TechNote for product evaluation. The performance of a complex computer system is highly dependent upon the tuning of the system and hardware subsystems. Do not base a purchase decision on synthetic benchmark results or vendor claims. Base it on the performance of a properly tuned system running a representative user workload. This TechNote explains how to properly tune a COMPAQ system for evaluation.

Performance Analysis Tools

Performance analysis tools are programs that capture "snapshots" of system activities and provide information to ensure efficient system operation. These tools include UNIX performance tools and third-party tools.

This section provides an overview of some of the UNIX performance analysis tools for monitoring system activities. For more information on these tools, see Chapter 5, "Performance Analysis Tools Reference."

The UNIX performance analysis tools are adequate to monitor most bottlenecks in system performance. Compared to third-party performance utilities, the shortcomings of the UNIX tools are mainly in data presentation. Performance monitoring tools from third-party vendors usually provide a graphical presentation of the performance parameters.

Compaq now offers a graphical performance analysis tool called **cpqmon**. You can find this tool on the COMPAQ Supplement for SCO UNIX diskette. The **cpqmon** tool uses curses(S) to display the system performance information such as disk I/O, internal system counters, and memory usage statistics.

NOTE: The COMPAQ Server Manager/R is another tool you can use to assess "system health." Server Manager/R provides "intelligent" system monitoring as well as comprehensive alerting and remote console emulation.

Capturing the Process State

The UNIX **ps** command produces information about active system processes. Entering **ps** without options produces information about system processes associated with the current user terminal. For system-level performance analysis, use the **ps** command with the **-e**, **-l**, and **-f** options. To print a short list of information about all processes, use the **-e** option. To print a long list, use the **-l** option. To print a full listing, use the **-f** option.

The following table compares the short and long listing options:

ps -e (Short Listing)	ps -l (Long Listing)
Process ID	All four items in the Short Listing, <i>plus</i> :
tty Identifier	Process State
Cumulative Execution Time	Process Status
Command Line	Parent Process ID
	Priority and Nice values
	Size of Core Image of Process
	Event for which Process is Waiting

The system administrator can manually invoke the **ps** command from the console, any terminal, or through a shell script written by the administrator. Because performance analysis usually requires capturing system-activity data over a period of time, consider incorporating the **ps** command in a data-gathering script that saves the data in a file for later analysis. For example, you can perform a **ps** data capture whenever you take a snapshot of the system activities with **sar**, or take a **ps** snapshot at the beginning and end of the **sar** data-capturing period.

Capturing System Activities and Statistics

The **sar** utility is a valuable UNIX tool for monitoring system activities. Many **sar** options are available with different ways to initiate them. The following sections describe various options as a part of specific recommendations for identifying subsystem bottlenecks. For more information on **sar**, see Chapter 5, "Performance Analysis Tools Reference."

The **sar** utility has two parts: data collection and data reporting. The **sadc** program collects the data. The UNIX daemon, **cron**, runs the */usr/lib/sa/sa1* script as a default after booting to a multiuser state. This script runs the **sadc** program.

The **sar** utility places the binary data collected by **sadc** into the file */usr/adm/sa/sadd* (where *dd* is the day of the month). When you run **sar** without the time interval option, **sar** goes to one of these files and reports the data on the specified system activity. For example, if you enter the **sar** command, the system displays all the collected system activity data for the day. Use this feature to produce an initial system activity report.

For a better system performance analysis, run **sar** in separate data collection and analysis steps. For data collection, use **sadc** with the following options:

```
/usr/lib/sa/sadc t n ofile
```

Where:

t = the measurement interval in seconds, usually 5 or greater

n = the monitor interval

ofile = the output file name

For example:

```
/usr/lib/sa/sadc 10 100 /usr/tmp/mysar.data
```

In this example, **sadc** starts to collect all system activity data in the background and puts the binary data in the */usr/tmp/mysar.data* file. This process occurs every 10 seconds for 100 times (about 17 minutes).

We recommend a **sar** monitoring interval of five to ten seconds or more. An interval shorter than five seconds might put too much load on the system and distort your view of the system activity.

After the data gathering is complete, you can use **sar** in the following form to print and analyze the data:

```
sar -ubdycwaqvmprDSAC -s xxxx -e yyyy -i zz -f filename
```

Where:

-ubdycwaqvmprDSAC = the activity options

xxxx = the starting time

yyyy = the ending time

zz = the interval in seconds

filename = the input data filename

For example:

```
sar -uq -s 14:00 -e 14:20 -i 10 -f mysar.data > textfile
```

In this example, **sar** extracts the data for system processor use and queue statistics from the *mysar.data* file. The data in *mysar.data* was collected every ten seconds from 14:00 to 14:20 (2:00 p.m. to 2:20 p.m.). The **sar** utility then writes the data in text format to the file *textfile*.

Use **sar** for quick system activity snapshots that the console displays. This is beneficial for evaluating current system conditions or examining the effects of running a specific process.

Listing the Kernel Parameters

You can use the **sysdef** command to display the current values of the kernel parameters. You can also use the System Administration shell, **sysadmsh**, to display and modify individual kernel parameters.

Chapter 2

Tuning the Memory Subsystem

This chapter describes the causes of bottlenecks in memory subsystem performance and presents steps for tuning the memory subsystem to enhance system performance. The chapter discusses the following topics:

- Causes of memory subsystem performance bottlenecks
- Identifying memory subsystem performance bottlenecks
- Guidelines for tuning the memory subsystem

Causes of Memory Subsystem Performance Bottlenecks

The memory subsystem becomes a performance bottleneck when the memory used by the kernel plus the memory used by the working sets of the active user processes approach the total physical memory capacity. Kernel memory requirements include the drivers, system tables, and buffers. The working set of a process is the set of pages the process recently referenced.

Not Enough Available Free Memory

When the available free memory in the system is below a low-water mark (defined by the kernel parameter **GPGSLO**), the page-stealer daemon starts to swap out aged pages to the swap device until the free system memory is above a high-water mark (**GPGSHI**). A moderate amount of page-out activity is generally acceptable. However, system performance deteriorates when page-out starts and gets progressively worse if the free-memory page count situation does not improve.

Critical Memory Shortage and System Performance

The most serious memory subsystem bottleneck occurs when the sum of the working sets of all processes exceeds the total physical memory available. The page-fault handler cannot allocate pages for a process, and the page-stealing daemon cannot steal pages fast enough because all pages are in the working sets.

When the kernel cannot allocate pages for a process, the swapper starts to swap out entire processes. Process swapping is slow due to the large amount of disk and system processor overhead. Eventually, the swapped-out processes must be swapped back in, causing more overhead.

When a critical memory shortage situation occurs, user response time is affected first. The swapper first tries to swap out the inactive processes. Because processes directly associated with the keyboard are inactive most of the time, the interactive performance deteriorates. In addition, the overall system performs poorly due to the high disk and system processor overhead. If the swap devices are not large enough, applications might fail to run.

As a general rule, for systems with *less* than 64 megabytes of memory, have swap space that is one and one half to two times the size of real memory. For systems with 64 megabytes or more, have a swap space that is at least equal to the amount of memory.

Identifying Memory Subsystem Performance Bottlenecks

To understand the memory subsystem performance, monitor the following system factors and activities:

- Memory allocation
- Virtual memory requirements for user processes and working sets
- Paging and swapping activities

Determining Memory Allocation

Determine the amount of memory used by the kernel and the amount of memory left for the daemons and user processes. This information is available from the boot messages in the `/usr/adm/messages` file. The information can help you analyze memory usage and tune kernel parameters.

NOTE: Boot messages are appended to the end of the `/usr/adm/messages` file.

For example, at the end of the `/usr/adm/messages` file you find the following:

```
mem: total = 16384k, kernel = 2716k, user = 13668k
```

In this case, there are 16 megabytes of memory. The kernel uses 2716 Kbytes, and 13668 Kbytes are available for user programs.

In certain situations, you might want to change some kernel parameters with **sysadmsh** and rebuild the kernel to make more memory available to the user. After modifying the kernel parameters and rebuilding the kernel, compare the kernel and user memory figures between the current and last boot messages to obtain the memory savings. When you install a new driver, you can also calculate how much memory the driver uses by this method.

.....

2-4 Tuning the Memory Subsystem

To find the amount of memory used for I/O buffers, view the most current boot messages in the `/usr/adm/messages` file.

For example, in the `/usr/adm/messages` file you find the following:

```
kernel: i/o bufs = 1736K
```

In this case, 1736 Kbytes are reserved for the system buffer cache.

This information is helpful when tuning the buffer cache to optimize disk and memory performance. The value of the I/O buffers is included in the kernel value.

As a default, the operating system determines how much system memory is present and uses this to determine a size for the buffer cache. As a general rule, try to have a buffer cache equal to 10 percent of memory. This value gives you a guideline to determine if you have enough memory for the user or if you must increase the buffer cache size in the future.

Monitoring User Processes Virtual Memory

During periods of poor system performance, try to discover the active processes and the system resources consumed by them. For memory subsystem bottlenecks, try to find out which processes are using the most memory, potentially creating the bottleneck.

Use the UNIX `ps` command with the `-el` options to take a snapshot of the system. This command produces a report that lists the memory requirements and active processes. The `-el` options report displays the following column headings:

```
F S UID PID PPID C PRI NI ADDR1 ADDR2 SZ WCHAN TTY TIME CMD
```

The "SZ" column of the report displays the size of the data and stack in 1-Kbyte blocks for each process. To find out the size of shared text, use the UNIX command **size**, and determine the size of the program file. This command provides the text, data, and bss (uninitialized data) sizes. Calculate the virtual memory requirement by adding the shared text size to the data and stack sizes for each invocation.

NOTE: The **size** utility is a component of the SCO UNIX Development System and is not found in the base operating system.

The **ps** command reports only the virtual memory requirement for each active process, not the working set requirement (which represents the actual memory consumed at that instance). However, the **ps** command does give some indication of overall memory consumption.

Monitoring Swap and Paging Activities

The most accurate indications of a bottleneck in the memory subsystem are the system swap and paging activities. The **sar** command, along with the **-r**, **-w**, and **-p** options, reports the following:

- Amount of free memory and space left on the swap device size (**-r** option)
- System swap activities (**-w** option)
- System paging activities (**-p** option)

NOTE: For more information on the **sar** command and options, see Chapter 5, "Performance Analysis Tools Reference."

When system performance is poor, run the **sar** command with **-rwp** options. After **sar** captures the system activities, analyze the reports and look for potential memory bottlenecks. We recommend a **sar** monitoring interval of five to ten seconds or more. An interval shorter than five seconds might put too much load on the system and distort your view of the system activity.

Finding the Amount of Free Memory

The **-r** option of the **sar** command reports the amount of free memory for each monitoring interval. The **-r** option report displays the following column headings:

freemem	freeswp
---------	---------

The "freemem" column shows the amount of free memory for user processes in 4-Kbyte pages. To find the values of the system parameters, **GPGSLO** (page-stealing low-water mark) and **GPGSHI** (page-stealing high-water mark), use the **sysdef** command. If the reported free memory is consistently lower than the high-water mark parameter, the system has a memory shortage.

The "freeswp" column reports the amount of free swap device blocks available. Each block is 512 bytes. This column indicates swap activity. If the values in this column vary throughout the monitoring period, there is swap activity.

You can also use the command **/etc/swap** to examine, add, or delete swap space. When you execute **swap** with the listing option **-l**, the system displays the path of the swap device(s), the "swaplow" value in 512-byte blocks, the size in blocks, and the number of free blocks. An example is shown below:

path	dev	swaplo	blocks	free
/dev/swap	1,41	049152	44544	

Investigating System Swap Activities

The **-w** option of the **sar** command reports the system swap activities. The **-w** option report displays the following column headings:

swpin/s	bswin/s	swpot/s	bswpot/s	pswch/s
---------	---------	---------	----------	---------

The "swpin/s" and "bswin/s" columns show the average number of swap-ins per second and the average number of disk blocks transferred per second during the swap-ins. These averages include swap-ins during normal demand paging through the page fault handler and normal process swap-in activities. By themselves, these numbers do not indicate a memory shortage problem.

To identify a memory bottleneck, check the "swpot/s" and "bswpot/s" columns. They report the average number of swap-outs per second and the number of disk blocks transferred per second during each swap-out. If both of these numbers are 0, no swap-outs occurred during the interval and the system does not have a memory bottleneck. If swpot/s is occasionally between 0 and 1, swap-outs of idle processes are occurring normally and there should not be a performance problem.

If all three of the following conditions exist, there are excessive swapping activities due to memory shortage:

- The swpot/s column is consistently greater than 1.
- The **-q** option report shows an average swap queue size (swpq-sz) of more than 2.
- The **-q** option report shows an average percentage occupancy of the swap queue (%swpocc) greater than 20 percent.

Investigating System Paging Activities

When you use the **sar** command with the **-p** option, the system reports paging activities. The **-p** option report displays the following column headings:

vflt/s	pflt/s	pgfil/s	rclm/s
--------	--------	---------	--------

The "vflt/s" column reports the average number of address translation faults per second. In a demand paging system, a validity fault occurs when the reference page was not in a process working set. A working set is a set of all pages referenced by a process within a default time. Because the processes are first loaded into memory through the page-fault handler, the vflt/s column by itself does not necessarily indicate a memory bottleneck.

The "pgfil/s" column reports the number of vflt/s per second satisfied by a page-in from the filesystem. However, as page fills approaches validity faults, there is a potential memory shortage.

The "rclm/s" column reports the average number of valid pages in the working set (currently active) that the system reclaimed and added to the free list by page-out activities. If rclm/s is consistently greater than 0 and the swap activity report shows swap-out activities, there is a memory shortage.

The "pflt/s" column reports the average number of protection faults per second and does not indicate a memory performance problem.

Example 1

The following **sar** reports are from a moderately loaded system, with only a small sample of monitoring intervals shown. The **-r**, **-w**, and **-p** options report memory-related activities. The **-u** and **-q** options report system processor loading statistics.

sar -u Option Report

Time	%usr	%sys	%wio	%idle
14:37:16				
14:37:26	6434	1	0	
14:37:36	333532	0		
14:37:46	11386	0		
14:37:56	75142	0		
14:38:06	53660	0		
14:38:16	92	8	0	0

sar -q Option Report

Time	runq-sz	%runocc	swpq-sz	%swpocc
14:37:16				
14:37:26	1.8	100		
14:37:36	1.5	40		
14:37:46	1.0	40		
14:37:56	1.5	40		
14:38:06	2.0	40		
14:38:16	1.2	100		
14:38:26	1.7	60		

The **-u** and **-q** option reports show that:

- The system had no idle time (%idle was 0).
- The average run queue size (runq-sz) was about 1.5.
- The percentage of time the run queue was occupied (%runocc) was about 40 percent.

.....

2-10 Tuning the Memory Subsystem

These figures indicate that the system was *not* bound by the system processor.

The large waiting-for-I/O (%wio) average of about 50 percent indicates that the system processor spent most of the time waiting for disk I/O to complete. The swap queue was always empty, both swpq-sz and %swpcc had no values, and there was no swapping due to memory shortage.

sar-r Option Report

14:37:16	freememfreeswp
14:37:26	25831608
14:37:36	26631608
14:37:46	27031608
14:37:56	32331608
14:38:06	30231608
14:38:16	39331608
14:38:26	47531608

The **-r** option report shows that the free memory average (freemem) was about 300. This figure considerably exceeds the system water-mark parameters of 25 and 40 pages, as reported by **sysdef**. There was enough free swap space (freeswp). Because there was no swapping, these values remained constant. Again, this lack of swapping confirms that there was no memory shortage.

sar-w Option Report

14:37:16	swpin/s	bswin/s	swpot/s	bswpot/s	pswch/s
14:37:26	0.00	0.0	0.00	0.0	42
14:37:36	0.00	0.0	0.00	0.0	62
14:37:46	0.00	0.0	0.00	0.0	128
14:37:56	0.00	0.0	0.00	0.0	59
14:38:06	0.00	0.0	0.00	0.0	65
14:38:16	0.00	0.0	0.00	0.0	8
14:38:26	0.00	0.0	0.00	0.0	37

The **-w** option report shows that there were no swap-ins or swap-outs during the example intervals; swpin/s, bswin/s, swpot/s, and bswot/s are all zero. Again, this confirms that there was no memory shortage which caused swapping. There was a great deal of *process* switching (pswch/s was around 60) but this does not indicate a memory bottleneck.

sar-p Option Report

14:37:16	vflt/spflt/spgfil/srclm/s
14:37:26	31.2032.000.000.00
14:37:36	36.0026.000.000.00
14:37:46	0.00 0.000.000.00
14:37:56	15.4012.200.000.00
14:38:06	26.0029.806.360.00
14:38:16	0.40 0.000.000.00
14:38:26	52.8077.004.600.00

The **-p** option report shows that the average number of pages reclaimed (rclm/s) was equal to zero throughout the reporting intervals. This system did not have a memory shortage problem.

Example 2

The following **sar** reports are from a heavily loaded system with a simulated workload of 64 users in a software development environment. Only a small sample of monitoring intervals are shown. The **-r**, **-w**, and **-p** options report memory-related activities. The **-u** and **-q** options report system processor loading statistics.

sar-u Option Report

15:05:35	%usr	%sys	%wio	%idle
15:05:45	7	22		700
15:05:55	6	23		710
15:06:05	1	17		820
15:06:15	5	34		600
15:06:25	4	24		720
15:06:35	3	24		720
15:06:45	3	15		830

sar-q Option Report

15:05:35	runq-sz	%runocc	swpq-sz	%swpocc
15:05:45	1.6	84	10.9	100
15:05:55	1.7	85	14.9	101
15:06:05	1.4	70	12.8	100
15:06:15	4.0	80	13.6	100
15:06:25	2.9	100	14.5	100
15:06:35	1.5	80	12.3	100
15:06:45	1.4	100	9.0	100

The **-u** and **-q** option reports show that the most important parameters to analyze are average swap queue size (swpq-sz) and the percentage occupied (%swpocc). During these intervals, the average swpq-sz was 12, and %swpocc was 100 percent. The system experienced a large amount of swapping during the monitoring intervals. The %wio averaged a very high 72 percent, probably due to the excessive amount of swapping. The excessive swapping was possibly caused by memory shortage.

sar-r Option Report

15:05:35	freememfreeswp
15:05:45	6222288
15:05:55	2321960
15:06:05	1821928
15:06:15	1421800
15:06:25	3421792
15:06:35	3221968
15:06:45	2121672

The **-r** option report shows that the free memory average (freemem) was below 40. This is low enough to dip below the system water-mark parameters and cause the system to swap. Note also that the amount of free swap space varied according to the amount of swapping. This relationship further confirms the swapping activities.

sar-w Option Report

15:05:35	swpin/s	bswin/s	swpot/s	bswpot/s	pswch/s
15:05:45	3.92	31.4	3.50	28.0	180
15:05:55	2.20	17.6	6.30	50.4	155
15:06:05	4.60	36.8	1.30	10.4	253
15:06:15	6.00	48.0	0.00	0.0	219
15:06:25	1.30	10.4	3.80	30.4	138
15:06:35	4.20	33.6	2.10	16.8	144
15:06:45	3.02	24.2	3.98	31.8	146

From the **-w** option report, the most important parameter to analyze is the average amount of swap-outs. In this case, swpot/s was about 3. Also, the swap-in activities in the last example were zero. Here, there were several swap-ins because pages that were swapped-out had to be swapped-in again.

sar-p Option Report

	vflt/s	pflt/s	pgfil/s	rclm/s
15:05:35				
15:05:45	5.80	4.00	1.50	2.80
15:05:55	11.20	9.80	4.20	2.40
15:06:05	12.30	9.80	2.70	0.00
15:06:15	13.30	26.70	7.00	1.10
15:06:25	9.01	6.04	2.54	2.89
15:06:35	11.70	12.33	5.12	2.78
15:06:45	8.90	10.70	3.40	3.09

In the **-p** option report, the parameter to look for is the average number of pages reclaimed through paging (rclm/s). Any value over zero indicates excessive swapping and paging. In this case, the average for rclm/s was around 2.

In this example, the excessive swapping (caused by memory shortage) definitely contributed a large part of the system performance problem. The disk overhead and increased system processor overhead were the main reasons for the decline in system performance.

Guidelines for Tuning the Memory Subsystem

This section covers recommended steps for tuning the memory subsystem to improve system performance. These steps include:

- Adding memory
- Balancing the system workload
- Tuning the kernel parameters

Balancing the system workload and tuning the kernel parameters promotes smooth and efficient daily system operation. However, the amount of system memory that you can gain from these procedures is usually small. These procedures are effective only if the system encounters *occasional* memory shortages. If the system is consistently short of memory, add more memory to improve performance. Then balance the workload and fine-tune the kernel parameters for the best overall performance.

Adding Memory

When critical memory shortage occurs consistently, the most effective solution is to add memory. Adding memory is the simplest and most important step to improve system performance. The additional tuning guidelines described in this section usually result in only marginal memory gain. In fact, some tuning procedures are performance trade-offs that might not work well during sustained system use.

To determine how much memory to add, first analyze the workload during periods when memory bottlenecks affect the system performance. Estimate the kernel memory requirement, the memory used by each active process, and the shared memory area (if any) allocated for each process. Use the system memory allocation data and the memory usage information from the `/usr/adm/messages` file and from the `ps -el` and `size` commands to plan how much additional memory you require.

NOTE: The `size` utility is a component of the SCO UNIX Development System and is not found in the base operating system.

The easiest way to resolve memory performance problems is to add memory until you have successfully removed the bottleneck condition. However, when you add memory you might also need to increase the swap device size.

After adding memory, tune the kernel parameters to improve overall system performance. For example, increase the size of the system buffer cache (**NBUF**). If you do not specify the **NBUF** parameter, the kernel automatically allocates a buffer cache at system startup. This default buffer cache might be larger than necessary for efficient disk performance. Use the techniques in this TechNote to determine the correct buffer cache size. Then specify the **NBUF** parameter equating to this size. In this manner, you will not waste memory due to an inefficient default allocation.

Example 3

The previous example discussed a performance problem caused by a bottleneck in the memory subsystem. In the following example, we added memory to the same system after running the **sar** program in a simulated workload for a 64-user, software-development environment. We obtained the following **sar** option reports:

sar-u Option Report

09:18:11	%usr	%sys	%wio	%idle
09:18:21	20	27	53	0
09:18:31	26	24	50	0
09:18:41	22	71	7	0
09:18:51	14	28	58	0
09:19:01	13	39	48	0
09:19:11	21	32	47	0
09:19:21	19	25	55	0

sar-q Option Report

	runq-sz	%runocc	swpq-sz	%swpocc
09:18:11				
09:18:21	19.2	50		
09:18:31	1.5	20		
09:18:41	18.4	90		
09:18:51	1.7	30		
09:19:01	1.5	20		
09:19:11	3.8	50		
09:19:21	2.7	30		

From the **-u** and **-q** option reports, the most important parameters were the average swap queue size and occupancy rate. After adding memory, there were no swapping activities, no value for either swpq-sz or %swpocc, and no swap activities due to memory shortage.

The system had no idle time (%idle was 0), but %wio was still high, averaging 45 percent. The average run queue size (runq-sz) was high, but the percentage of time the run queue was occupied (%runocc) was low, at less than 50 percent. The added memory probably shifted the performance bottleneck to the disk subsystem. The system then held up the processes while it waited for I/O to complete.

sar-r Option Report

09:18:11	freemem	freeswp
09:18:21	492	31608
09:18:31	485	31608
09:18:41	420	31608
09:18:51	332	31608
09:19:01	382	31608
09:19:11	394	31608
09:19:21	416	31608

The **-r** option report shows that the free memory average (freemem) was about 400. This average far exceeds the system water-mark parameters. There was enough free swap space (freeswp). Because there was no swapping, the freeswp values remained constant, confirming that no swapping due to memory shortage occurred.

sar-w Option Report

	swpin/s	bswin/s	swpot/s	bswpot/s	pswch/s
09:18:11					
09:18:21	0.00	0.0	0.00	0.0	180
09:18:31	0.00	0.0	0.00	0.0	241
09:18:41	0.00	0.0	0.00	0.0	306
09:18:51	0.00	0.0	0.00	0.0	130
09:19:01	0.00	0.0	0.00	0.0	322
09:19:11	0.00	0.0	0.00	0.0	180
09:19:21	0.00	0.0	0.00	0.0	258

The **-w** option report shows that there were no swap-ins or swap-outs during the example intervals (swpin/s, bswin/s, swpot/s, and bswpot/s were all 0).

sar-p Option Report

09:18:11	vflt/s	pflt/s	pgfil/s	rclm/s
09:18:21	0.90	5.10	0.30	0.00
09:18:31	12.90	18.40	3.70	0.00
09:18:41	2.00	4.80	0.30	0.00
09:18:51	17.20	13.50	1.80	0.00
09:19:01	0.00	0.00	0.00	0.00
09:19:11	4.70	7.70	0.90	0.00
09:19:21	0.00	0.00	0.00	0.00

The **-p** option report shows that the average number of pages reclaimed (rclm/s) was back to 0 throughout the monitoring period. The return to 0 confirms that no critical swapping and paging occurred.

These **sar** option reports confirm that the additional memory eliminated the memory bottleneck. These reports also show that the system performance bottleneck probably shifted from memory to the disk subsystem. The steps to identify a bottleneck in the disk subsystem and the related tuning guidelines are in Chapter 3, "Tuning the Disk Subsystem."

Balancing the System Workload

When you suspect a memory bottleneck, run the **sar** and **ps** commands over a period of time. This captures system activities and helps you determine when the memory bottlenecks occur and what processes cause them. You can execute these commands either periodically through the **cron** daemon or directly from the system console.

Rescheduling workloads around busy periods might avoid memory bottlenecks that consistently occur only during certain periods of the day. When certain combinations of user workloads cause memory bottlenecks, request that users modify their system-usage patterns. Avoid administrative and cleanup programs during these periods by rearranging the **crontab** table entries.

Tuning the Kernel Parameters

You can increase the memory available to user processes by tuning kernel parameters. The gain might be marginal. However, if you previously configured the system with excessive buffers and unused stream buffers, you might gain significantly more memory.

Tuning the kernel parameters sometimes causes a tradeoff of kernel resources among competing system components. A change in the kernel parameters to improve the performance of one component might adversely impact the performance of another. Also, changing kernel parameters requires relinking the kernel and rebooting the system to make the new parameters effective. This disrupts daily system operations.

We recommend that you carefully analyze the bottleneck situations and establish a performance baseline before making any kernel parameter changes. Then make your kernel parameter changes in small, experimental steps.

The following tables provide information and tuning guidelines on several kernel parameters.

Table 2-1
Paging Parameters

Parameter	Comments and Guidelines
AGEINTERVAL	The number of clock ticks a process runs before its pages are aged. The default is 19. If the system is chronically short of memory, decrease AGEINTERVAL to make more memory pages available to active processes. However, decreasing AGEINTERVAL usually has a negative effect on interactive performance.
GPGSLO	The low-water mark in pages. Normally, value for GPGSLO is about 1/16 of pageable memory. Default value is 25. Increase value to make daemon more active; decrease value to make daemon less active. Value must be a positive integer smaller than value of GPGSHI. If you increase the value of GPGSLO you must also adjust GPGSHI.
GPGSHI	The high-water mark of free memory in pages. Normally, value for GPGSHI is about 1/10 of pageable memory. Default value is 40. Increase value to make daemon more active, decrease value to make daemon less active. Value must be a positive integer greater than GPGSLO, and usually not greater than 10 megabytes.

Table 2-2
Buffer Cache Parameters

Parameter	Comments and Guidelines
NBUF	The number of 1-Kbyte buffers in buffer cache at boot time. If you do not specify a value for NBUF, the system calculates a default size at system startup. Size is listed in the file <i>/usr/adm/messages</i> . Decreasing NBUF frees up more memory, but might negatively affect disk performance. Study <i>sar -b</i> report for information on cache hit rate. Cache hit rate increases with number of buffers. When most processes are compute-bound or consistently access vary large files, decrease NBUF to about 5 percent of total memory.
NHBUF	The number of hash queues to allocate for 1-Kbyte buffers. Value must be a power of 2 and chosen so NBUF divided by NHBUF is approximately equal to 4.

**Table 2-3
STREAMS Parameters**

Parameter	Comments and Guidelines
NBLK n	Controls the number of STREAMS data blocks and buffers allocated for each size class. Numbers also used to allocate message block headers. The number of message blocks is 1.25 times total of all data block allocations.
NQUEUE	The number of STREAMS queues. Queues always allocated in pairs. Minimal STREAM contains four queues. Each module pushed on a STREAM requires two additional queues. Typical configuration value for systems not running X applications is 4 times NSTREAM.
NSTREAM	The number of STREAMS head structures. Recommended value is highly application dependent, but a value between 32 to 40 usually suffices for running single transport provider with moderate traffic.

**Table 2-4
Multiscreens Parameters**

Parameter	Comments and Guidelines
NSCRN	The maximum number of multiscreens. A value of 0 configures this value at boot time based on amount of installed memory. Maximum value is 12.
SCRNMEM	The number of 1024-byte blocks for console screen saves. A value of 0 configures this value at boot time based on amount of installed memory. Maximum value is 128. When using a non-zero value, make SCRNMEM equal to 6 times NSCRN.

Table 2-5
Kernel Tables Parameters

Parameter	Comments and Guidelines
NPROC	The number of process tables to allocate. Value should be between 50 and 3000. Use the sar -v report to compare number of used tables entries to the number allocated. Set NPROC to a value that is roughly 15 percent higher than the largest value in <i>proc-sz</i> column of the sar -v report.
MAXUP	The number of concurrent user processes a non-super user may run. Set value of MAXUP at least 10 percent smaller than value of NPROC .
NFILE	The number of open file table entries to allocate. Normal value in range of 100 to 600. NFILE must be less than or equal to NINODE .
NMOUNT	The number of mount table entries to allocate. The <i>root</i> filesystem is always the first entry. Set the value for NMOUNT as low as possible. The system searches the mount table linearly.
NREGION	The number of region table entries to allocate. Can be set between 100 and 10,000. As a general rule, set NREGION to a value that is slightly more than three times the value of NPROC .
NINODE	The number of inode table entries to allocate. Normal value in range of 100 to 400. NINODE must be greater than or equal to NFILE .
NCALL	The number of call-out table entries to allocate. Value must be in the range of 30 to 500. As a general rule, set NCALL to a value slightly more than the number of enabled tty devices.
NCLIST	The number of character list buffers to allocate. The average number needed per terminal is in the range of 5 to 10.

Decreasing the value of these kernel tables parameters decreases the size of the system tables and increases the memory available for user processes. Use the **sar -v** command to compare the average size of the inode table, process table, and file table with the allocated sizes. If the percentage is small, consider reducing the table size. The amount of acceptable adjustment depends on the system operating environment.

NOTE: Generally, the memory gained from reducing the table sizes is small, and over-adjustments might cause the system to fail when a process exceeds the system-resource limits.

Chapter 3

Tuning the Disk Subsystem

This chapter describes the causes of disk I/O bottlenecks and presents steps for tuning the disk subsystem to enhance system performance. This chapter discusses the following topics:

- UNIX filesystem I/O
- Identifying disk subsystem performance bottlenecks
- Guidelines for tuning the disk subsystem

NOTE: This chapter pertains only to the COMPAQ SMART SCSI Array Controller (SMART Controller). This chapter does not address the performance of the COMPAQ Intelligent Drive Array Controller (IDA) or the COMPAQ Intelligent Drive Array (IDA-2) Controller-2.

UNIX Filesystem I/O

To execute disk I/O activities through the UNIX filesystem, the user processes access data on the disks by making "read and write" system calls. For read system calls, the UNIX kernel searches through the buffers in the hash queue of the buffer cache. If the kernel finds the data in the buffer cache (a "cache hit"), no physical disk read is necessary. To acquire data that is not found in the buffer cache (a "cache miss"), the kernel initiates a block-read request to the SMART Controller through the device driver.

.....

3-2 *Tuning the Disk Subsystem*

For write system calls, the kernel stores the data in a buffer rather than sending a write request immediately to the controller. At regular intervals, or as necessary, the kernel sends all accumulated buffers to the device driver. The device driver then requests the controller to perform the disk write function.

For UNIX filesystems, factors contributing to disk subsystem performance are the system buffer cache, the IDA device driver, and the SMART Controller. For example, a poorly configured buffer cache might result in a cache-miss amount of more than 20 percent. This would cause the system to perform excessive physical disk I/O.

Raw Device I/O

Some applications optimize disk I/O performance by using raw devices. These applications bypass the disk buffer cache and directly access the data from raw devices. For example, most database management systems set up the table spaces in raw devices and furnish their own buffer cache scheme optimized for accessing data tables. In this instance, the factors contributing to the disk subsystem performance are the application program data block management system, the IDA device driver, and the SMART Controller.

COMPAQ SMART SCSI Array Controller

The COMPAQ SMART SCSI Array Controller optimizes disk throughput by incorporating advanced features such as bus mastering, data striping, parallel data transfers, simultaneous request servicing, optimized request management, and a four-megabyte cache. With these features, the controller can:

- Handle simultaneous disk I/O requests from the user processes.
- Manage the requests to optimize performance.
- Send data from the drive array in parallel to achieve a throughput that is theoretically equal to the combined throughput of each drive.
- Cache write requests to the disks, providing near-zero latency writes. This feature is especially helpful in the case of raw device I/O and synchronous writes through the filesystem (for example, the case of an NFS server) where a process must wait for the data transfer to the disk media before continuing execution.

There might still be instances when the disk subsystem cannot keep up with the number of requests from the user processes. A bottleneck occurs when requests from the device driver come to the SMART Controller faster than the controller can process them. Additional disk requests are placed ("queued") in the device driver, and are sent to the SMART Controller later. Meanwhile, outstanding requests sent from the device driver are also queued in the SMART Controller. When the SMART Controller becomes saturated with requests, disk performance degrades in proportion to the number of outstanding disk requests.

Identifying Disk Subsystem Performance Bottlenecks

This section describes steps to identify disk subsystem bottlenecks. The discussion focuses on the types of disk activities to monitor, how to use the UNIX **sar** utility to monitor disk activities, and how to interpret the **sar** output to identify potential disk bottlenecks.

There are three basic types of disk activities to monitor:

- Block I/O
- Swap and paging
- Raw disk I/O

Block I/O

Block I/O is buffered disk I/O through the buffer cache when the user process makes a read or write system call. The two types of activities associated with block I/O are logical I/O and physical block I/O. A logical I/O operation occurs when the request is satisfied without accessing the physical disk (for example, reading from the buffer cache). Physical block I/O occurs when access to the physical disk is required to satisfy the request.

Swap and Paging

Swap and paging occur in normal system activities such as:

- Process start-up
- Demand paging
- Critical swap and paging activities due to severe memory shortage

Swap and paging initiate raw disk I/O rather than disk I/O channeled through the system buffer cache.

Raw Disk I/O

Raw disk I/O activities bypass the system buffer cache. Therefore, system buffer cache configuration is not a factor. These kinds of disk I/O benefit most from the write-posting cache of the SMART Controller. Raw disk I/O can occur during one of the following activities:

- System backup to a raw disk partition
- Database management system engines that bypass the filesystem and buffer cache, directly accessing database and data table spaces in raw disk partitions

Monitoring Percentage Waiting for I/O

The UNIX **sar -u** command reports system processor use and activities.

The **-u** option report displays the following column headings:

```
%usr%sys%wio%idle
```

The "%usr" column is the percentage of time that the system processor is running in user mode, and "%sys" is the percentage of time it is running in system mode. The "%wio" column shows the percentage of system processor time consumed waiting for block I/O. The "%idle" column shows the time the system processor is idle. If %wio is consistently greater than 20 percent, the system is spending a large percentage of time waiting for disk I/O to complete. This can be a potential disk bottleneck.

NOTE: For more information on the **sar** command and options, see Chapter 5, "Performance Analysis Tools Reference."

Example 1

The following **sar -u** options report is from a simulated workload of several users doing a large amount of disk I/O activities in a software development environment:

Time	%usr	%sys	%wio	%idle
14:22:16				
14:22:26	3	27	69	0
14:22:36	40	22	38	0
14:22:46	9	46	45	0
14:22:56	1	30	69	0

During the intervals that statistics were taken, the system spent an average of 50 percent of the processor resource waiting for disk I/O to complete. System performance suffered from this bottleneck in the disk subsystem.

Example 2

Analyzing the %wio alone sometimes cannot reveal a disk bottleneck situation. %wio is an aggregate, system-wide percentage. Analyze %wio together with data from the following **sar** command options:

- The **-q** option for the run queue size statistics
- The **-b** option for block I/O buffer statistics
- The **-c** option for the system call statistics

The following **sar -u** option report is from a simulated workload of 64 users in a software development environment:

Time	%usr	%sys	%wio	%idle
14:46:10				
14:46:20	23	77	0	0
14:46:30	27	73	0	0
14:46:40	29	71	0	0
14:46:50	23	77	0	0

The time intervals in this report show that the system spent zero percent of the time waiting for disk I/O to complete. The report shows there was considerable disk activity.

If we use the **-q**, **-b**, and **-c** option reports to investigate the run queue and block I/O statistics during the same intervals, we see why the %wio number does not tell the complete story on this disk I/O bottleneck.

sar -q Options Report

The **-q** option report displays the following column headings:

runq-sz	%runocc	swpq-sz	%swpocc
---------	---------	---------	---------

The "runq-sz" column provides the average number of processes in the run queue that are ready to run. The "%runocc" column represents the average percentage of time the run queue is occupied. The "swpq-sz" and "%swpocc" columns are the swap queue of processes to be swapped out and the percentage of time the swap queue is occupied.

Example 2 Continued

	runq-sz	%runocc	swpq-sz	%swpocc
14:46:10				
14:46:20	53.8	100		
14:46:30	54.4	100		
14:46:40	39.4	100		
14:46:50	33.4	100		

The run queue statistics from the **-q** option show that numerous processes were ready to run and the run queue was occupied 100% of the time. For a detailed description of this option, refer to Chapter 5, "Tuning the System Processor."

You can also use statistics from the **-b** and **-c** options to report a large amount of physical disk I/O. The **-b** and **-c** option reports and examples are described in a later section of this chapter.

The %wio number from the **-u** option report was zero because the system was so overloaded it did not have time to wait for I/O. In this situation, system performance suffered from *both* a disk system bottleneck *and* a system processor bottleneck.

Monitoring Buffer Block I/O and Raw I/O

The COMPAQ SMART Array Controller supports the **sar -d** option. This option allows you to query the device driver for information about the drive array activity level. For more information on the **sar -d** option report, see Chapter 5, "Performance Analysis Tools Reference."

You can also use the **-b** and **-c** options of the **sar** command to interpret the activity level of the controller, as follows:

1. Apply a known disk workload with benchmark programs.
2. Measure the average number of block I/O (which is equal to the sum of average block reads and writes per second plus the physical reads and writes per second).

Monitoring Buffer Cache and Block I/O Activities

The UNIX **sar -b** option report shows buffer cache and block I/O activities. The **-b** option report displays the following column headings:

```
bread/s  lread/s  %rcache  bwrit/s  lwrit/s  %wcache  pread/s  pwrit/s
```

The "%rcache" and "%wcache" columns measure the buffer cache hit ratios. If %rcache is below 90 or %wcache is consistently below 65, there are not enough buffers in the pool to cache the disk I/O efficiently. In this case, increase the buffer cache size by modifying some kernel parameters to minimize the number of accesses to the physical disk.

The "bread/s," "bwrit/s," "pread/s," and "pwrit/s" columns represent the average number of physical block I/Os per second. The bread/s column shows the average number of "block reads" from the disk subsystem to the buffer cache. The bwrit/s column shows the average number of "block writes" from the buffer cache to the disk subsystem. Together, breads/s and bwrit/s portray buffer-cache physical block I/O activities. The pread/s and pwrit/s columns report the average number of raw-device block reads and writes per second due to raw disk I/O activities.

The sum of the bread/s, bwrit/s, pread/s, and pwrit/s columns represents the average number of physical disk I/Os to the SMART Controller. This sum indicates the activity level of the controller.

The following are indications of a possible bottleneck in the disk subsystem:

- The %rcache is below 90 percent or the %wcache is below 65 percent.
- The disk subsystem has an array configured with no fault tolerance, and the sum of bread/s, bwrit/s, pread/s, and pwrit/s is greater than 30 times the number of drives per array (for example, 120 for a 4-drive array).
- The disk subsystem has an array of two physical drives with no fault tolerance, and the sum of bread/s, bwrit/s, pread/s, and pwrit/s is above 40.

NOTE: When the SMART Controller is configured in fault tolerance mode, you may lower the thresholds 20 percent for disk mirroring (RAID 1) or distributed parity (RAID 5), and 40 percent for data guarding (RAID 4). For more information on fault tolerance, see the section, "Disk Access Patterns and Fault Tolerance Selection" found later in this chapter.

These guidelines are very general. Other factors such as logical drive partitioning, number of filesystems, size and distribution of the filesystems, filesystem fragmentation, and the access patterns (sequential or random) play an important role in overall disk system performance.

Monitoring Block I/O System Call Activities

Although the **sar -c** option report is not directly related to disk subsystem activity, it provides additional information that might be helpful in the analysis. The **-c** option report displays the following column headings:

scall/s	sread/s	swrit/s	fork/s	exec/s	rchar/s	wchar/s
---------	---------	---------	--------	--------	---------	---------

The "sread/s" and "swrit/s" columns report the average number of read and write system calls. In a heavy disk I/O situation, these entries might represent about half of the total system calls.

The "rchar/s" entry (the total number of bytes transferred by read system calls per second) divided by sread/s produces the average size of the read system calls.

The "wchar/s" column (the total number of bytes transferred by write system calls) divided by swrit/s produces the average size of the write system calls.

Example 3

The following **sar** reports were generated with a workload simulating a 64-user software development environment. The **sar** command captured all the system activities for a period of time. The following example shows **-b** and **-c** option reports for a few selected intervals. These intervals are the same as those discussed in the previous section.

sar -b Option Report

14:46:10	bread/s	lread/s	%rcache	bwrit/s	lwrit/s	%wcache	pread/s	pwrit/s
14:46:20	34	2133	98	40	363	89	0	0
14:46:30	39	1957	98	40	384	90	0	0
14:46:40	34	1250	97	42	434	90	0	0
14:46:50	44	1787	98	41	358	89	0	0

sar -c Option Report

	scall/s	sread/s	swrit/s	fork/s	exec/s	rchar/s	wchar/s
14:46:10							
14:46:20	2253	1480	55	6.00	6.60	263948	45266
14:46:30	1656	859	53	8.80	9.20	159069	39140
14:46:40	1536	517	58	8.40	8.40	177370	42572
14:46:50	1585	860	51	6.00	6.20	176131	42430

Because pread/s and pwrit/s equal zero, there was no raw device I/O. All physical disk I/O was channeled through the system buffer cache. The cache hit ratio was about 98 percent for reads and about 89 percent for writes. The system buffer cache was not a performance problem. However, the sum of all the physical disk I/O averages was consistently greater than 75, indicating a severe disk system bottleneck.

The **sar -c** option report shows that there were many more reads than writes during these intervals. The average number of characters was about 115 per read call and 82 per write call. This information can be useful in determining the size of reads and writes to raw partitions.

Example 4

The following **sar** reports with **-b**, **-c**, **-q**, and **-u** options were generated with a workload simulating a database transaction processing environment. In this kind of application, the database data and table spaces are usually configured on raw devices, and the database process manages its own disk buffer cache.

sar -b Option Report

14:46:18	bread/s	lread/s	%rcache	bwrit/s	lwrit/s	%wcache	pread/s	pwrit/s
14:46:28	0	59	100	0	59	100	60	59
14:46:38	0	55	100	0	55	100	55	52
14:46:48	0	60	100	0	60	100	60	59
14:46:58	0	60	100	0	60	100	60	58

sar -c Option Report

14:46:18	scall/s	sread/s	swrit/s	fork/s	exec/s	rchar/s	wchar/s
14:46:28	988	120	120	0.00	0.00	180913	142250
14:46:38	903	111	109	0.00	0.00	168875	131441
14:46:48	976	122	121	0.00	0.00	180870	146713
14:46:58	962	120	119	0.00	0.00	180781	141913

sar - q Option Report

Time	runq-sz	%runocc	swpq-sz	%swpocc
14:46:18				
14:46:28	10.1	100		
14:46:38	10.5	100		
14:46:48	10.6	100		
14:46:58	9.2	100		

sar - u Option Report

Time	%usr	%sys	%wio	%idle
14:46:18				
14:46:28	61	38	1	0
14:46:38	66	33	1	0
14:46:48	60	38	1	0
14:46:58	66	32	2	0

During these selected reporting intervals, there were about the same number of read and write system calls (sread/s and swrit/s). The small lread/s and lwrit/s numbers indicate very few reads and writes from and to the system buffer cache. There were virtually no physical reads or writes through the buffer cache. The cache hit ratio for both reads and writes was about 100 percent for the example intervals and about 99 percent throughout the test.

There was, however, a large number of raw device I/Os. The sum of pread/s and pwrit/s was consistently over 100. Because the sum of all the physical I/O was above 100 per second, this indicates a severe disk system bottleneck.

There is also an indication of a marginal system processor bottleneck. As shown in the **sar -u** and **-q** option reports, the average run queue size was about 10, and the average system processor use was close to 100 percent.

Guidelines for Tuning the Disk Subsystem

This section describes procedures to improve disk subsystem performance. If you perceive a performance problem, go through the diagnostic steps and verify that the problem is a disk subsystem bottleneck. Then use some or all of the following guidelines to tune the disk subsystem performance:

- Understanding disk access patterns and fault tolerance selection
- Adding an additional SMART Controller
- Creating logical drives
- Limiting disk fault tolerance
- Tuning the kernel buffer cache parameters
- Maintaining efficient filesystems

Some guidelines require changes to the disk configuration. You will obtain the most performance gain from these changes. Changing the disk configuration effectively requires a good understanding of the user community and the applications.

Disk Access Patterns and Fault Tolerance Selection

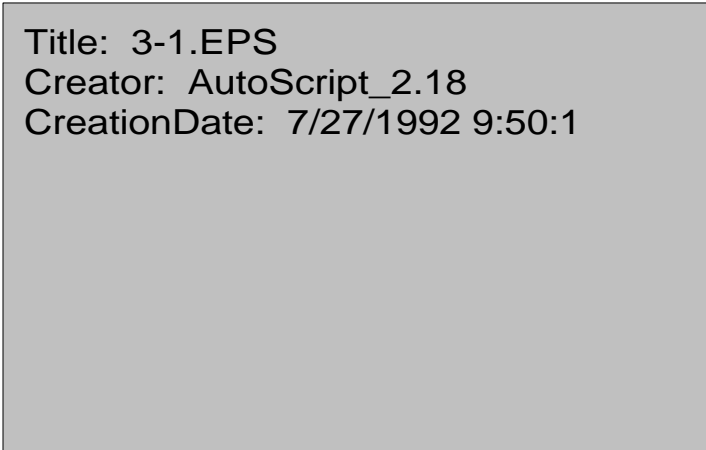
The SMART Controller manages up to 14 disks (seven disks on each of its two integrated SCSI channels). To obtain optimal performance with the SMART Controller, you must understand the I/O data access pattern to the SMART Controller and how this affects the choice of a controller fault tolerance scheme.

- Data striping with no fault tolerance (RAID 0)

Sequential I/O performance improves in a nearly linear fashion up to four disks per SCSI channel. (Data throughput continues to improve up to seven disks per SCSI channel. However, the gain in throughput for each additional disk drops off after four disks per channel because the SCSI bus begins to saturate). Random I/O performance scales very well up to the maximum 14-disk configuration.

- Drive Mirroring (RAID 1)

RAID 1 provides drive mirroring, where one drive controller duplicates all write operations to two sets of independent hard drives.



Title: 3-1.EPS
Creator: AutoScript_2.18
CreationDate: 7/27/1992 9:50:1

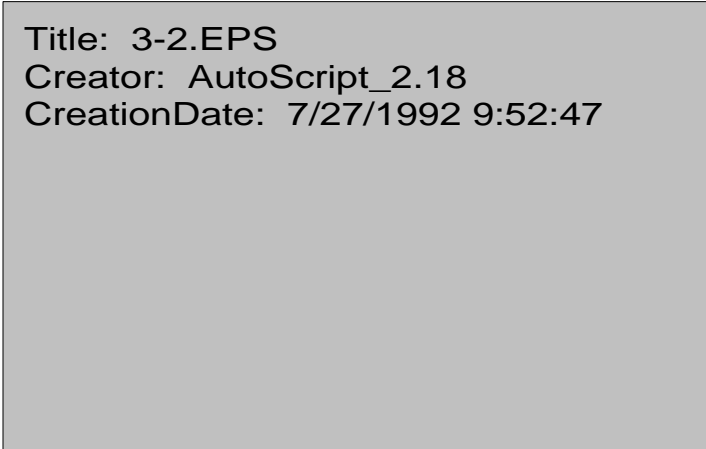
Figure 3-1. Drive Mirroring

Drive mirroring reduces the capacity of a logical volume by 50 percent. Read performance on the array approaches that of RAID 0 (no fault tolerance) because both copies of the logical volume can be accessed independently. Write performance, however, is reduced to that of a RAID 0 system with half the number of drives, because copies of data must be written to both mirrored sets of drives simultaneously.

In general, RAID 1 provides the best performance of the fault tolerance choices on the SMART Controller. However, this performance comes at the expense of half the available disk space.

- Data guarding (RAID 4)

RAID 4 provides data guarding, where the drive controller uses a dedicated hard drive (called the "parity drive") to store encoded data.



Title: 3-2.EPS
Creator: AutoScript_2.18
CreationDate: 7/27/1992 9:52:47

Figure 3-2. Data Guarding

The logical volume capacity is only reduced by one disk/spindle, contrasted to the 50-percent capacity sacrificed in RAID 1. Read performance on the array equals that of a RAID 0 array with one fewer disk. Write performance depends on the rate and size of the write requests.

The cache on the SMART Controller ensures near-zero "write latencies" as long as the rate of writes to the SMART Controller does not overflow its ability to flush data to the disks. Because all parity information is kept on a single spindle, only one write can be performed simultaneously on the array, once the cache fills. Thus RAID 4 performance, in a worst-case scenario, approaches that of a single disk/spindle.

In general, RAID 4 is well suited to applications characterized by a very high read-to-write ratio (90:10) or better, and/or applications that perform very large block transfers (greater than 64-KByte blocks).

- Distributed parity (RAID 5)

RAID 5 provides distributed parity, where the drive controller distributes encoded parity data across all disks in the array (versus a single spindle in RAID 4).

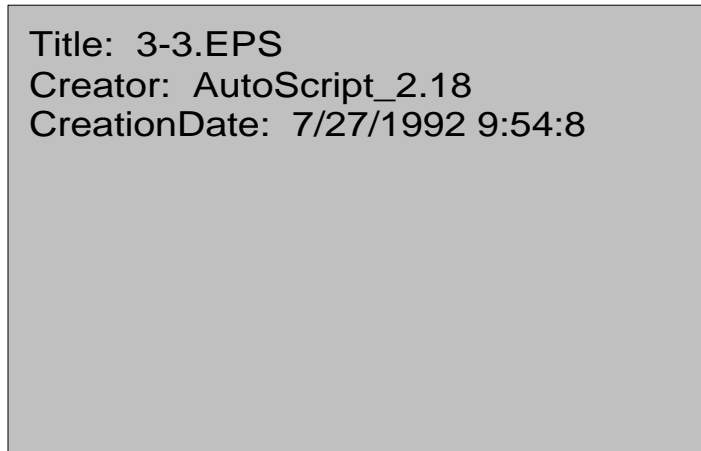


Figure 3-3. Distributed Parity

Again, this reduces the logical volume capacity by only one disk/spindle, compared to the 50-percent capacity sacrificed in RAID 1.

Read performance on the array equals that of a RAID 0 array with one less disk. Write performance depends on the rate and size of the write requests. The cache on the SMART Controller ensures near-zero write latencies as long as the rate of writes to the SMART Controller does not overflow its ability to flush data to the disks. Once the cache fills, the RAID 5 implementation achieves a steady-state performance nearly equal to RAID 1 (mirroring). This is because each write requires two disks to fulfill (data and parity) and thus the effective write rate is reduced to (at best) half of a RAID 0 (no fault tolerance) configuration.

In general, RAID 5 is well suited to applications characterized by a high read-to-write ratio (75:25) or better, and performing writes with a block size smaller than the array's distribution factor (block size of 16 KBytes or smaller, so the SMART Controller can perform these in parallel at two disks per I/O).

Adding an Additional SMART Controller

If the current configuration contains more than eight physical drives (four on each SCSI channel) and the workload is characterized by highly sequential I/O, you can improve the disk subsystem performance by adding another SMART Controller and distributing the workgroup filesystems and raw devices among logical drives between the two SMART Controllers.

By spreading out the filesystems and raw devices, you avoid disk contention and maximize throughput and performance. Try to distribute filesystems from workgroups or processes that compete for disk subsystem resources over logical drives on these separate SMART Controllers.

Creating Logical Drives

Before reconfiguring the SMART Controller, analyze the filesystem requirements of the user workgroups. If the majority of user I/O operations are:

- Sequential and in large blocks (such as program loads or database search and scan) - you can optimize performance by configuring the controller as a single logical volume with multiple filesystems.
- A mixture of random and sequential accesses (for example, one group of users primarily loading programs, and other groups of users accessing database tables as raw devices) - it might be more advantageous to configure the controller as two or more logical drives (maximum of eight).

For example, if the SMART Controller has *eight* attached drives, you can create two logical drives: one logical drive composed of six disks for database or other random I/O workloads, and one logical drive composed of two disks for highly sequential access files/filesystems. If the SMART Controller has only *four* attached drives, then the preferred setup would likely involve a single logical volume with separate filesystems for sequential and random I/O.

The following figure illustrates these two situations:



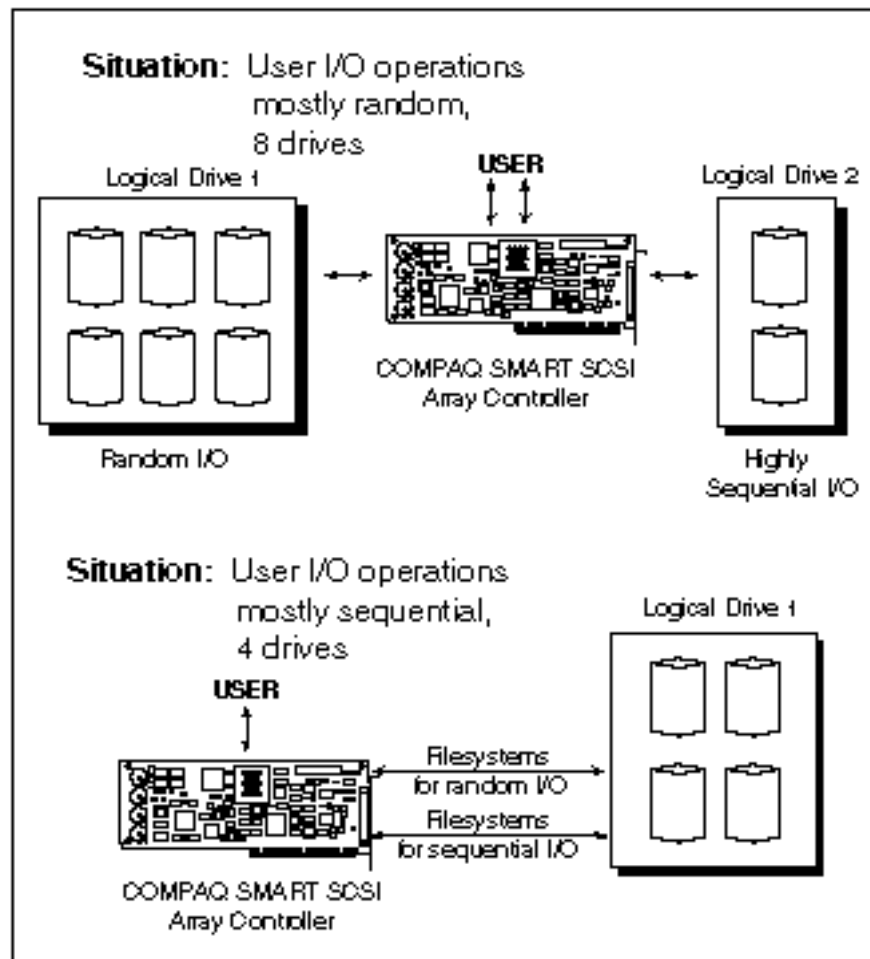


Figure 3-4. Logical Drive Example

The COMPAQ EISA Configuration Utility allows you to specify the logical drive configuration for the SMART Controller. You can partition the disk and create filesystems under UNIX with the **fdisk** and **divvy** commands.

Limiting Disk Fault Tolerance

RAID 1, RAID 4, and RAID 5 provide fault tolerance at the expense of disk subsystem performance and capacity. RAID 1 reduces the available storage capacity by 50 percent with little impact on disk performance. RAID 4 and RAID 5 reduce the storage capacity by only one physical disk, which is used to store the parity information, but incur a potential performance penalty, compared to RAID 1.

RAID 5 performance on the SMART Controller is characterized by the ratio of read-to-write accesses and the distribution of disk accesses. If applications are characterized as highly read intensive (greater than 75:25 read-to-write ratio), and the writes occur in short bursts that do not overrun the 2-megabyte cache on the controller, then RAID 5 performance approaches that of RAID 1. Write throughput can be as much as 50 percent less than with no fault tolerance.

Tuning the Kernel Buffer Cache Parameters

If the buffer cache read and write hit ratios (%rcache and %wcache - reported by the **sar -b** option report) are consistently below 90 and 65 percent respectively, you can improve system performance by tuning the kernel buffer cache parameters. The buffer cache size is listed in the file */usr/adm/messages*.

Buffer cache size is related to memory problems, such as insufficient system memory or reduced buffer cache size (to improve memory performance). In either case, a solution might be to add more memory and increase the buffer cache size.

To increase the buffer cache, use **sysadmsh** to tune the kernel parameters. The following table provides comments and tuning guidelines:



Table 3-1
Kernel Buffer Cache Parameters

Parameter	Comments and Tuning Guidelines
NBUF	The number of 1-Kbyte buffers in buffer cache at boot time. If you do not specify a value for NBUF, the system calculates a default size at system startup. You can find the size of NBUF in the <i>/usr/adm/messages</i> file in the format: kernel: i/o bufs = xxxK. NBUF is usually around ten percent of the physical memory. For optimal performance, increase the number of hash queues (NHBUF) along with NBUF (see MAXBUF, below).
NPBUF	The number of physical I/O buffers to allocate. You need one I/O buffer for each active physical read and write.
NHBUF	The number of hash queues to allocate for the 1-Kbyte buffers. Value must be a power of 2 and chosen so NBUF divided by NHBUF is approximately equal to 4.
MAXBUF	The maximum number of buffers in the buffer cache. When you adjust NBUF, adjust MAXBUF accordingly.
BDFLUSHR	The rate, in seconds, for checking the need to write the filesystem buffer to the disk. The default is 30. Increasing BDFLUSHR decreases disk activity, at the expense of reliability.
NAUTOUP	The "age," in seconds, that a delayed-write buffer must obtain before BDFLUSHR writes it out. The default is 10. Increasing NAUTOUP keeps buffers in memory longer, but increases the risk of data loss if the system goes down.

Maintaining Efficient Filesystems

To administer your system properly, practice the following maintenance programs:

- Maintain small filesystems.
- Maintain free space in the filesystem.
- Reduce filesystem disk fragmentation.
- Keep directories small.
- Tune the name-cache and scatter-gather I/O.

Maintaining Small Filesystems

As a general rule, a UNIX filesystem performs better when it is small. We recommend that you create a root filesystem and additional small filesystems for different workgroups of the user community. The average occupancy rate of these small filesystems should be around 85 percent. This configuration ensures efficient filesystems and prevents the root filesystem from running out of space.

Maintaining Free Space in the Filesystem

A UNIX system operates best when at least 15 percent of the space of each filesystem is free. When the free space in an active filesystem is less than 15 percent, its performance becomes sluggish. If a filesystem is full, the system stops any user attempt to write to it, causing the process to "hang." To prevent this, monitor the filesystem free space regularly and take preventive measures.

When filesystem free space falls below 15 percent, perform the following actions:

1. Ask all users of the filesystem to remove their unneeded files.
2. Locate large directories and files in the filesystem, and ask the owners to remove their unneeded files.
3. Remove any temporary, old, core, and log files.
4. Make a complete backup of the filesystem, remove all non-critical files, and restore these from the backup as needed.
5. If necessary, create and mount an additional filesystem.

Reducing Filesystem Disk Fragmentation

After a filesystem is in use for a time, the constant creation and deletion of files scatters the free space over the logical drive. Newly created files are likely to be scattered in small blocks. This condition hinders the SMART Controller performance features (such as disk synchronization and striping) that are designed to enhance the disk subsystem throughput.

No tools are currently available to diagnose disk fragmentation for the SMART Controller. Therefore, take periodic measures to reduce filesystem disk fragmentation.

If you suspect filesystem fragmentation, use the **sysadmsh** utility to back up all files in that filesystem, delete all the files in the filesystem, and then restore from the backup. This action generally frees about five to ten percent of the filesystem space. Because the restored files are written sequentially and contiguously, the restored filesystem can take full advantage of the SMART Controller performance features.

Keeping Directories Small

For the best system performance, keep the filesystem directories as small as feasible. Directory lookup is relatively slow, and a directory will not shrink. Deleted files are marked in the directory, but directory space is not reclaimed. As time goes on, there are more and more directory entries.

A directory with an average file length name of 14 allows as many as 30 file entries to fit into a single disk block. Such a directory can be searched very efficiently.

A directory can have as many as 286 entries and still be viable, as long as the directory is only for data storage. A working directory containing more than 286 file entries generally causes poor performance.

Prior to SCO UNIX version 3.2.4, a file name directory entry was 16 characters in length, 2 to store the inode, and 14 for the file name. Version 3.2.4 uses variable-length file names.

To count the number of inode entries in the directory, enter the following command:

```
hd . | wc -l
```

NOTE: The count is off by one. Subtract one to correct.

The solution to the large directory problem is twofold: educate users to keep their login directories small, and compress the directories by using **cpio** to back up and restore files in the directories.

Tuning the Name-Cache Parameters

The names of recently used files are cached. This reduces the time the system spends connecting a filename with its contents. If files are accessed multiple times, system efficiency increases. If the entries hit% from the **sar -n** report is below a 90-percent cache hit rate, increase the caching parameters.

Some of the tunable name-cache parameters are listed in the following table:

Table 3-2
Name-Cache Parameters

Parameter	Definition
S5CACHEENTS	Number of name components in name cache.
S5HASHQS	Number of hash queues for name cache.
S5OFBIAS	Bias towards keeping open files in cache.

To tune the cache parameters, follow this procedure:

1. Make **S5CACHEENTS** equal to **NINODE** times 3.
2. Make **S5HASHQS** a prime number roughly equal to **S5CACHEENTS** divided by 4.
3. Set **S5OFBIAS** to **S5CACHEENTS** divided by 10.

Tuning the Scatter-Gather I/O Parameters

The Acer Fast Filesystem (AFS) gathers together groups of disk I/O requests, rather than sequentially executing individual requests. This increases the overall efficiency of disk operations. The tunable parameter **NMPBUF** controls the number of buffers used to gather large disk requests before transferring to user space or the buffer cache. **NMPHEADBUF** allocates stand-alone cluster buffer headers.

When you are configuring fast filesystem buffers, there is a tradeoff between memory use and disk subsystem performance. We recommend that you set **NMPHEADBUF** to 16 and leave **NMPBUF** at 0 so it is auto-configured at boot time.

Chapter 4

Tuning the System Processor

This chapter describes the causes of bottlenecks in system processor performance and presents steps for tuning the system processor to enhance performance. This chapter discusses the following topics:

- System processor activities
- Identifying system processor performance bottlenecks
- Guidelines for tuning the system processor

System Processor Activities

There are four kinds of system processor activities:

- Running in user mode
- Running in kernel mode
- Waiting for I/O
- Idle

System processor use is the percentage of time that the system processor is either in user mode executing code which is not part of the kernel, or in kernel mode executing kernel code. A well-balanced system processor usually spends about the same amount of time running in user and kernel mode, and spends about a third of the time waiting for I/O or in an idle state.

Identifying System Processor Performance Bottlenecks

The system processor becomes a performance concern when both the processor use and load average are high. System processor use is the percentage of time the system processor is executing either user or kernel processes; it does not include the percentage of time the processor is idle or waiting for I/O. The system load average is the average number of processes in the run queue ready to run during a specific interval. When the system processor use is consistently over 80 percent for the overall system or over 60 percent for time-response-critical workloads, and the system load average is consistently over 5, the system processor is a performance bottleneck.

System processor bottlenecks increase the time to run user processes. When the time increase becomes unacceptable to the users, tune the system processor.

To help identify a performance problem with system processor, monitor the following system factors and activities:

- System load average
 - Percentage of run queue occupied
 - Percentage of system and user mode
 - Percentage idle time
 - System workload
-

Monitoring the System Load Average

To see a report on the system load average, use the UNIX **uptime** command. A high load average means increased system processor load. A load average of five is considered heavy.

The **uptime** command reports the load average in three intervals: last minute, last five minutes, and last 15 minutes. Use this command as a quick-trend indicator for the average system load over a long period of time. For a more in-depth analysis, **sar** is a better tool. For example, the following output from the **uptime** command indicates a lightly loaded system with a slightly increasing load over the last 15 minutes.

```
2:32pm up 2 days 10:20 , 4 users, load average 1.2, 1.3, 1.5
```

The load average from **uptime** also counts the processes waiting for disk I/O to complete. Correlate this information with the %wio number from the **sar -u** option report. For more information on the **uptime** command, see Chapter 5, "Performance Analysis Tools Reference."

Monitoring the Percentage of System Processor Use

Use the Corollary Multiprocessor Activity Monitor, **mpstat**, to monitor the percentage of system processor use for a system with one or more system processors. The utility graphically displays the percentage of system- and user-mode processor activity for each active system processor. The **mpstat** tool is especially useful in a multiprocessor system because **sar** reports the overall, combined processor use.

.....

4-4 Tuning the System Processor

NOTE: Starting with SCO MPX 3.0, the **mpsar** command can report **sar** data for each processor in a multiprocessor installation. The **mpsar** command uses most of the same flags as **sar**. For more information on **mpsar**, see Chapter 5, "Performance Analysis Tools Reference."

When the system processor activity graphs are consistently over 80 percent, the system processor is a bottleneck. In the dual-system processor configuration, if the first system processor is consistently over 80 percent used and second processor is only marginally used, there could be a load-balancing problem.

A typical source of a load-balancing problem is heavy use of a non-multiprocessor subsystem. In SCO UNIX, there is only streams-based networking (TCP/IP is not yet multiprocessor capable). If load-balancing is not the problem, check any third-party device drivers and ensure they are multi-threaded.

The UNIX **sar** command with **-u** and **-q** options reports the system processor use percentage, the run queue statistics, and the swap queue statistics. When viewed together, this information can indicate a potential system processor bottleneck.

The **-u** option of the **sar** command reports the percentage of time the system processor is spending in each of four activities. The **sar -u** option report displays the following column headings:

%usr	%sys	%wio%idle
------	------	-----------

The "%usr" column is the percentage of time running in user mode and "%sys" is the percentage of time running in system mode. Together, these columns indicate the percentage of system processor use. Over a period of time, if the combined %usr and %sys percentage is over 80 percent, the system is processor-bound. However, this number alone is usually *not* enough to identify a system processor bottleneck. You must also analyze the system load average from the **uptime** command and the run queue statistics with the **-q** option of the **sar** command.

The "%wio" column is the time the system processor spent waiting for blocked I/O. This number is a *disk* bottleneck indicator. If this number is high, the system is disk-bound.

The "%idle" column shows the time the system processor is idle. In theory, the most efficient system would have almost zero idle time. From the standpoint of system performance, you must accept a few percentage points of idle time to ensure adequate system processor response time.

The **-q** option of the **sar** command produces reports that list run queue and swap queue activities. The **sar -q** option report displays the following column headings:

runq-sz	%runocc	swpq-sz	%swpocc
---------	---------	---------	---------

The "runq-sz" column provides the average number of processes in the run queue that are ready to run. This number represents the system processor workload. As a general rule, if the runq-sz number is greater than 5, there is a potential system processor bottleneck. The recommended value of 2 is probably too conservative. An average run queue size of less than 5 is generally acceptable.

The "%runocc" column represents the average percentage of time the run queue is occupied. Higher numbers indicate higher system processor use. A very high %runocc does *not* necessarily indicate a bottleneck; it indicates a problem only when runq-sz is also high.

The "swpq-sz" and "%swpocc" columns can indicate memory bottlenecks. Although excessive swapping also increases system processor and disk overhead, high numbers here can mean that the system performance problem is due to a bottleneck in memory.

Over a period of time, if the %usr number *plus* the %sys number is greater than 80, *and* the runq-sz number is greater than 5 for a single-processor system or 7 for a dual-processor system, there is a system processor bottleneck.

For more information on the **sar** command, see Chapter 5, "Performance Analysis Tools Reference."

Example

The following **sar** reports with the **-u**, **-q**, **-w**, and **-b** options are from a system with a simulated workload that mixes system processor and floating-point intensive scientific application programs with a moderate amount of editing, compilation, and file-management activities. Only a small sample period taken from the report is used in this example.

sar-u Option Report

14:32:53	%usr	%sys	%wio	%idle
14:33:03	95	5	0	0
14:33:13	92	8	0	0
14:33:23	92	8	0	0
14:33:33	95	5	0	0
14:33:43	88	12	0	0
14:33:53	99	1	0	0
14:34:03	99	1	0	0
14:34:13	89	11	0	0

sar-q Option Report

14:32:53	runq-sz	%runocc	swpq-sz	%swpocc
14:33:03	7.0	100		
14:33:13	6.7	100		
14:33:23	7.3	100		
14:33:33	6.8	100		
14:33:43	6.6	100		
14:33:53	7.1	100		
14:34:03	7.2	100		
14:34:13	6.3	100		

From the **sar -u** option report, the sum of the percentage of system processor time in user mode (%usr) plus the percentage of time in system mode (%sys) equaled 100 percent throughout the sample period. Both the percentage of time waiting (%wio) and percentage of idle time (%idle) were zero. The system processor was loaded 100 percent of the time, and spent an average of over 90 percent of the time in the user mode. (%usr averaged about 94 percent). The system was loaded with compute-bound processes that made very few system calls (characteristic of computation-intensive scientific programs).

The **sar -q** option report shows that the average run queue size (runq-sz) was over 7 and the run queue occupancy rate (%runocc) was 100 percent. As a general rule, whenever the run queue size average is consistently over 5, you have a system processor bottleneck. Therefore, in this case the system performance undoubtedly suffered from a system processor bottleneck.

sar-w Option Report

14:32:53	swpin/s	bswin/s	swpot/s	bswpot/s	pswch/s
14:33:03	1.10	8.8	0.00	0.0	27
14:33:13	1.60	12.8	13.10	104.8	55
14:33:23	5.30	42.4	0.60	4.8	56
14:33:33	9.50	76.0	1.60	12.8	35
14:33:43	7.20	57.6	8.10	64.8	70
14:33:53	0.10	0.8	0.00	0.0	3
14:34:03	0.70	5.6	0.00	0.0	2
14:34:13	0.50	4.0	1.25	12.0	177

The **sar -w** option report shows there was a small amount of swap-out activities as indicated by the non-zero swap-outs average (swpot/s) for some interval. After examining these and additional intervals not shown in this example, we see that the overall averaged swpot/s was small. System memory was not yet a critical factor in the system performance, but adding memory would certainly improve the overall performance.

sar-b Option Report

	bread/s	lread/s	%rcache	bwrit/s	lwrit/s	%wcache	pread/s	pwrit/s
14:32:53								
14:33:03	2	108	99	10	66	84	0	0
14:33:13	8	62	86	1	11	88	0	0
14:33:23	15	90	84	1	22	95	0	0
14:33:33	5	29	84	1	5	86	0	0
14:33:43	9	160	94	9	94	90	0	0
14:33:53	0	0	100	0	0	100	0	0
14:34:03	0	1	100	0	1	100	0	0
14:34:13	81	172	53	6	65	90	0	0

The **sar -b** option report shows little disk I/O activity. The average number of physical I/O bread/s, bwrit/s, pread/s, and pwrit/s was very small. There were occasional spurts of disk I/O, but this would not be a performance concern. In this example, the disk subsystem was *not* a factor in the system performance.

Monitoring the System Workload

The **ps** command shows which processes are using the system processor resources and how much resource each process is using. First, use the **sar** command to analyze the system processor activities and run queue statistics. If you determine that there is a system processor bottleneck, look at the active processes with the **ps** command. The report from either the **ps -el** or **ps -ef** command can help you identify the processes consuming most of the system processor resource.

The **ps -el** report displays several column headings. However, the columns that are pertinent to system processor use are:

- S** The state of the process
- C** The system processor use percentage estimate for scheduling purposes
- PRI** The priority of the process
- TIME** The accumulated execution (system processor) time of the process

For more information on the **ps** command, see Chapter 5, "Performance Analysis Tools Reference."

Once you determine which processes are consuming most of the system processor resource, you can take one or more actions. See the section, "Guidelines for Tuning the System Processor."

Guidelines for Tuning the System Processor

This section describes actions you can take to tune the system processor performance. These actions include:

- Adding a second system processor board
- Balancing the workload mix
- Improving the efficiency of user-defined features
- Tuning the kernel parameters

Adding a Second System Processor Board

Before you add another system processor board, verify that system performance is not caused by bottlenecks in other system components. Analyze other system activities with the UNIX **sar** command.

If system performance is poor due to high system processor use *and* the processor is spending most of its time on processor-bound programs, the most effective solution is to upgrade the system to a dual-processor configuration.

Balancing the Workload Mix

When a system consistently runs out of system processor capacity and cannot handle some workload combination satisfactorily (as determined by the **sar** and **ps** commands), you might alleviate the performance problem by balancing the workload mix. Try to have a steady workload distribution throughout the day, rather than in bursts of heavy system workloads. For example, arrange the workload so that all heavy system processor-bound processes are not active at the same time.

Consider rescheduling maintenance and system activities to avoid periods of heavy work load. To reschedule, change the scripts of the **crontab** entries in the */usr/spool/cron/crontabs* directory.

Improving the Efficiency of User-Defined Features

Part of the workload redistribution effort is to request that the users avoid running specific programs during certain time periods. You can lower the priorities for system processor-intensive and non-interactive programs with the **nice** command. Although the programs take longer to run, the overall system response time improves.

An efficient \$PATH environment variable is important to performance. Every time you execute UNIX command, the system references the \$PATH variable. Directory lookups are linear and relatively slow, and access time increases as the number of directory entries increase. Follow these general guidelines:

- Keep \$PATH as short as possible with no duplicate directory names.
- List the directories referenced most often first, followed by the directories referenced less frequently (\$PATH is searched from left to right).
- Avoid large directories, or place them at the end of the \$PATH list (searching large directories is inefficient).
- List the current directory "." last for security reasons.

In the course of creating and deleting files, the physical directory increases in size. A directory never decreases in size. When you delete a file the directory entry for that file is marked as "free" and is available for use on the next file creation. Directory searches are most efficient when the directory size is 2Kbytes or less.

When a directory exceeds 2Kbytes in size, you can compress it and restore the search efficiency. To compress a directory, first create a temporary directory. Copy all the files in the original directory to this temporary directory. Delete the original directory. Rename the temporary directory to the original directory name.

Tuning the Kernel Parameters

Unnecessarily large kernel tables can slow down some kernel algorithms. This occurs because the algorithms search the tables linearly. Your initial system configuration might include default table sizes. The default table sizes represent an *anticipated* operating environment.

After the system is in operation and you have a better understanding of the environment, you can resize some of the kernel tables to improve performance. As a general rule, keep the kernel tables as small as possible without causing an application "crash" because the system runs out of table entries.

Use **sysdef** to view and **sysadmsh** to view and change kernel parameters. Some of the kernel parameters you can tune are listed in the following table:

Table 4-1
Kernel Parameters

Parameter	Comments and Guidelines
NPROC	The number of process tables to allocate. Value should be between 50 and 3000. Use the <code>sar -v</code> report to compare number of used tables entries to the number allocated. Set NPROC to a value that is roughly 15 percent higher than the largest value in <i>proc-sz</i> column of the <code>sar -v</code> report.
MAXUP	The number of concurrent user processes a non-super user may run. Set value at least 10 percent smaller than value of NPROC.
NFILE	The number of open file table entries to allocate. Normal value in range of 100 to 600. Value must be less than or equal to NINODE.

continued

Table 4-1 *continued*

Parameter	Comments and Guidelines
NMOUNT	The number of mount table entries to allocate. The <i>root</i> filesystem is always the first entry. Set the value as low as possible. The system searches the mount table linearly.
NREGION	The number of region table entries to allocate. Can be set between 100 and 10,000. As a general rule, set value slightly more than three times the value of NPROC.
NINODE	The number of inode table entries to allocate. Normal value in range of 100 to 400. NINODE must be greater than or equal to NFILE.
NCALL	The number of call-out table entries to allocate. Value must be in the range of 30 to 500. As a general rule, set NCALL to a value slightly more than the number of enabled tty devices.
NCLIST	The number of character list buffers to allocate. The average number needed per terminal is in the range of 5 to 10.

Decreasing these parameter values improves how efficiently the kernel processes the tables. The **sar -v** command reports the average percentage occupancy of some of the system tables. Use this report as a guideline for reducing the table sizes. If the percentage of table size is small, consider reducing the table size.

Chapter 5

Performance Analysis Tools

Reference

This chapter provides a reference to some of the commands used in the previous chapters for monitoring systems activities, statistics, and workload.

The PS Command

Purpose: Reports process status

Syntax: ps -el

Description: The **ps** command reports status information about active processes. Although there are many options for this command, the following are the most useful for system performance issues.

- e prints information about all processes.
- l Generates a long list including: status, priority, location, and memory usage information for each process.

Comments: The **ps** tool provides a "snapshot" of the status, memory requirements, I/O status, and the priority of the active processes. When a performance bottleneck occurs, this information can help you identify which system resource(s) each process consumes and to what degree.

Taking a **ps** snapshot might distort the system resource picture. The **ps** command uses resources of the system processor and memory. The **ps** output also tends to provide a narrow view of the system; it focuses on the *processes*, instead of on the system as a whole.

The following table provides detailed information on the column headings found in a **ps -el** report.

**Table 5-1
ps -el Report Information**

Report Column	Description
F	Status word flag with the values added to form a single octal value. Possible values are: 00process terminated 01a system process in primary memory 02parent is tracing process 04tracing parent signal stopped process, parent waiting 08process in primary memory 10process in primary memory and locked until event completes 20process cannot be swapped
S	Process state. Possible values are: 0process is running on processor Ssleeping; process waiting Rrunnable process on run queue lidle; process being created Zzombie; process terminated and parent not waiting Ttraced; process stopped, parent tracing it XSXBRK; process waiting for more primary memory
UID	User ID of the user who created the process.

continued

Table 5-1 *continued*

Report Column	Description
PID	Process identification number.
PPID	Parent process ID.
C	Percentage of system processor used for scheduling. A lower value means a higher priority. I/O-bound processes have low value. Compute-bound processes have a high value.
STIME	Process start time. If the process started in the last 24 hours, this value is the process start <i>time</i> . Otherwise, this value is the process start <i>date</i> .
PRI	Priority of the process. A higher number indicates lower priority. Priority is a function of system processor use, system base priority, and the nice value (NI).
NI	Process nice value.
ADDR1, ADDR2	Memory addresses of the u-area of the processes, if resident. Otherwise, these values are the <i>disk</i> addresses. ADDR1 is the frame address of the first half of the u-area. ADDR2 is the frame address of the second half.
SZ	Process core image size, in blocks. Value does <i>not</i> include shared text size, but <i>does</i> include stack size, and varies with stack size.
WCHAN	Address of the event counter that the process is waiting for. If this value is left blank, the process is running.
TTY	Terminal associated with the process.
TIME	Cumulative execution (system processor) time used by the process.
CMD	Command name.

The SAR Command

Purpose: Reports system activity

Syntax: sar[A-aBbcdghmnpqRruvwy] [-A] [-o *file*] *t* [*n*]
sar[A-aBbcdghmnpqRruvwy] [-A] [-s *time*] [-e *time*]
[-i *sec*] [-f *file*] /usr/lib/sa/sadc [*t n*] [*ofile*]

Description: In the first syntax instance shown above, **sar** samples cumulative activity counters in the operating system at *n* intervals of *t* seconds, where *t* should be 5 or greater. If the **-o** option is specified, **sar** saves the samples in a file in binary format. The default value of *n* is 1.

In the second syntax instance, with no sampling interval specified, **sar** extracts data from a previously recorded file, either the one specified by the **-f** option or, by default, the standard system activity daily data file */usr/adm/sa/sadd* for the current day (*dd*).

The starting and ending times of the report can be bounded via the **-s** and **-e** time arguments in the form *hh[:mm[:ss]]*.

The **-i** option selects records at *sec* second intervals. Otherwise, it reports all intervals found in the data file.

Comments: Any information that is displayed "per second" is the *average* over the interval *t*. Each value is calculated by taking the total number of occurrences of the event over the duration of the interval *t*, divided by the interval *t*.

The following table provides detailed information on the **sar** command options and the column headings. This information applies only to SCO UNIX release 3.2v4.



Table 5-2
sar Command Options and Report Information

Option	Description	Report Column Headings
-A	Summary of all reports	
-a	Reports use of file access system routines.	<i>iget/s</i> Number of files located by i-node entry per second. <i>namei/s</i> Number of filesystem path searches per second. <i>dirblk/s</i> Number of directory block reads issued per second.
-B	Reports additional buffer cache activity.	<i>cpybuf/s</i> Number of copy buffers required per second. <i>slpcpybuf/s</i> Number of times necessary to "sleep" while waiting for a copy buffer.
-b	Reports buffer cache activity.	<i>bread/s, bwrit/s</i> Average number of block reads or writes, per second, between system buffers and disk or other block devices. <i>lread/s, lwrit/s</i> Average number of logical block reads and writes, per second, from system buffers. <i>%rcache, %wcache</i> Percentage of logical reads and writes found in buffer cache. <i>pread/s, pwrit/s</i> Average number of physical reads and writes (block reads and writes), per second.
-c	Reports system calls.	<i>scall/s</i> System calls of all types, per second. <i>sread/s, swrit/s, fork/s, exec/s</i> Specific system calls, per second. <i>rchar/s, wchar/s</i> Characters transferred by read and write system calls, per second.

continued

Table 5-2 *continued*

Option	Description	Report Column Headings
-d	Reports activity for each block device,(disk or tape drive). When data displays, the device specification <i>dsk-</i> generally represents a disk drive. The device specification representing a tape drive is machine dependent.	<p><i>%busy, avque</i> Portion of time device was busy servicing a transfer request; average number of requests outstanding during that time.</p> <p><i>r+w/s, blks/s</i> Number of read and write transfers to device, per second; number of 512-byte blocks transferred to the device, per second.</p> <p><i>await, avserv</i> Average time, in milliseconds, that transfer requests wait idly on queue; average time, in milliseconds, to be serviced (for disks, this includes seek, rotational latency, and data transfer times).</p>
-g	Reports on serial I/O.	<p><i>ovsiohw/s</i> Overflows at sio hardware.</p> <p><i>ovsiodma/s</i> Overflows at sio dma cache.</p> <p><i>ovclist/s</i> Overflows of clists.</p>

continued

Table 5-2 *continued*

Option	Description	Report Column Headings
-h	Reports buffer statistics.	<i>mpbuf/s</i> Number of mp (scatter-gather) buffers allocated per second. <i>ompb/s</i> Number of times system ran out of mp buffers per second. <i>mphbuf/s</i> Number of mp buffer headers allocated per second. <i>omphbuf/s</i> Number of times system ran out of mp buffer headers per second. <i>pbuf/s</i> Number of physio buffers per second. <i>spbuf/s</i> Number of sleeps/s waiting for physio buffers per second. <i>dmabuf/s</i> Number of dma transfer buffers allocated per second. <i>sdmabuf/s</i> Number of sleeps/s waiting for dma transfer buffers per second.

continued

Table 5-2 *continued*

Option	Description	Report Column Headings
-m	Reports message and semaphore activities.	<i>msg/s, sema/s</i> Number of message and semaphore operations, per second.
-n	Reports name cache statistics.	<i>c_hits, cmisses</i> Number of name cache hits and misses. <i>hit%</i> Hit-to-miss ratio as a percentage.
-p	Reports paging activities.	<i>vflt/s</i> Number of address translation page faults (valid page not in memory), per second. <i>pflt/s</i> Number of page faults from protection errors (illegal access to page) or "copy-on-writes," per second. <i>pgfil/s</i> Number of <i>vflt/s</i> satisfied by page-in from filesystem, per second. <i>rclm/s</i> Number of valid pages reclaimed by the system, per second.
-q	Reports average queue length while occupied, and % of time occupied.	<i>runq-sz, %runocc</i> Run queue of processes ready to run; percentage of time swap queue is occupied. <i>swpq-sz, %swpocc</i> Swap queue of processes to be swapped out; percentage of time swap queue is occupied.

continued

Table 5-2 *continued*

Option	Description	Report Column Headings
-R	Reports on process activity.	<i>dptch/s</i> Number of times the dispatcher is run. <i>idler/s</i> Number of times the idler is run per second. <i>swtdle/s</i> Number of times idler is switched to per second.
-r	Reports unused memory pages and disk blocks.	<i>freemem</i> Average number of 4-Kbyte pages available to user processes. <i>freeswap</i> Number of 512-byte disk blocks available for process swapping.
-u	Reports system processor use (the default).	<i>%usr, %sys, %wio, %idle</i> Portion of time running in user mode, running in system mode, idle with some process waiting for block I/O, and otherwise idle.
-v	Reports status of selected kernel tables (process, inode, file, and lock tables).	<i>proc-sz, inod-sz, file-sz, lock-sz</i> Current number of table entries and table sizes. <i>ov</i> Number of times overflow occurred between sampling points for each table.

continued

Table 5-2 *continued*

Option	Description	Report Column Headings
-w	Reports system swapping and switching activity.	<i>swpin/s, swpot/s, bswin/s, bswot/s</i> Number of transfers into memory and number of 512-byte block units transferred for swapins and swapouts, per second (including initial loading of some programs). <i>pswch/s</i> Process switches, per second.
-y	Reports TTY device activity.	<i>rawch/s, canch/s, outch/s</i> Input character rate, input character rate processed by canon, output character rate, per second. <i>rcvin/s, xmtin/s, madmin/s</i> Receiver, transmitter, and modem interrupt rates, per second.

Data Gathering

The operating system contains several counters that increase when various system actions occur. These include counters for system processor use, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, and paging. The **sadc** and **shell** procedures (*sa1* and *sa2*) sample the data, save it, and process it.

sadc, the data collector, samples system data *n* times, with an interval of *t* seconds between samples, and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than five seconds; otherwise, the activity of **sadc** might affect the sample.

If *t* and *n* are omitted, a special record is written. At system boot time (when booting to a multiuser state), this facility marks the time when the counters restart from zero. For example, the */etc/init.d/perf* file writes the restart mark to the daily data by the command entry:

```
su sys -c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

The shell script *sa1*, a variant of **sadc**, collects and stores data in binary file */usr/adm/sa/sadd*, where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted.

The entries in */usr/spool/cron/crontabs/root*:

```
0 * * * 6,0 /usr/lib/sa/sa1 3600
```

```
0 8-17 * * 1-5 /usr/lib/sa/sa1 3600
```

```
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
```

produce records every 20 minutes during working hours and hourly otherwise.

The shell script *sa2*, a variant of **sar**, writes a daily report in file */usr/adm/sa/sardd*.

The */usr/spool/cron/crontabs/root* entry:

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

reports important activities hourly during the working day.

Exit Values

Upon successful completion, **sar** exits with 0. If you specify an invalid option, **sar** exits with value 2. It exits with value 1 for all other errors.

Diagnostic Messages

sar: Incompatible start and end times specified (*etime* <= *stime*)

The specified end time is the same as or before the start time.

sar: Time step and/or number of steps requested are invalid

The time interval or number of intervals specified are not integer values, are negative values, or are otherwise invalid.

sar: Can't open filename

The system cannot open the specified input file with the **-f** option.

sar: ofile same as ffile

The specified input and output files are identical.

sar: argument -- illegal argument for option

The argument specified for the option is invalid.

Command Examples

To see the current day's system processor activity:

```
sar
```

To watch system processor activity evolve for 10 minutes and save data:

```
sar -o temp 60 10
```

To review the disk and tape activity from that period at a later time:

```
sar -d -f temp
```

Warning

Running multiple copies of **sar** can affect the results. The kernel automatically extracts the data using **sar**. It is the extraction process, not the collection that consumes resources. Therefore, results produced when running multiple copies might not reflect the actual system performance.

The current version of **sar** is compatible with older versions of **sar**. Any data files saved with older versions can be read with the current version.

Files

/usr/bin/sar - command

/usr/adm/sa/sadd.num - daily data file

/usr/adm/sa/saradd.num - daily report file

/usr/lib/sa/sa.adrfl - address file

Standards Conformance

sa1, *sa2*, **sadc**, and **sar** are conformant with AT&T SVID Issue 2.

The MPSAR Command

Purpose: Reports system activity on MPX systems.

Syntax: `mpsar [-AaBbcdFghImnpQqRruvwy] [-A] [-ofile] t [n]`
`mpsar [-AaBbcdFghImnpQqRruvwy] [-A] [-s time]`
`[-e time][-i sec] [-f file]`

Description: You can use the **mpsar** command in conjunction with **cpusar** to report system activity on MPX (multiprocessor) systems. The **mpsar** command reports system activity for the entire MPX system. In the case of system processor usage and other data displayed per system processor by **cpusar**, **mpsar** displays the combined data for all system processors together. To report system activity for an individual system processor, use **cpusar**. For a description of the method used to gather data, see the **sar** manual page.

When you use **mpsar** with a time interval *t* the tool samples cumulative activity counters in the operating system at intervals of *t* seconds (where *t* should be five or greater). If you specify the value *n* (the number of intervals), the sampling repeats *n* times. If you do not specify *n*, it defaults to 1. If you specify the **-o** option, the system saves **mpsar** data in a binary format to the file.

If you do not specify a time interval, **mpsar** extracts data from a previously saved file. This file can be one specified using the **-f** option or, by default, the standard system activity daily data file `/usr/adm/sa/sadd`, where *dd* is the current day of the month. You can find the starting and ending times of the report by using the **-s** and **-e** arguments where time is specified in the form `hh[:mm[:ss]]`. The

-i option selects records at *sec* second intervals. If you do not specify **-i**, the system reports all intervals found in the data file.

If you specify **mpsar** without any of the options listed below, it displays the percentage of total system processor use, **-u**, by default. Any information that displays as "per second" value is the average over the interval *t*. Each value is calculated as the total number of occurrences of the event over the duration of the interval *t*, divided by the interval *t*.

The following table provides detailed information on the **sar** command options and the column headings found on each of the reports.

Table 5-3
mpsar Command Options
and Report Information

Option	Description	Report Column Headings
-A	A summary of all reports.	
-a	Reports use of file access system routines: <i>iget/s, namei/s, dirblk/s.</i>	
-B	Reports additional buffer cache activity.	<i>cpybuf/s</i> Number of copy buffers required per second. <i>slpcpybuf/s</i> Number of times necessary to "sleep" waiting for a copy buffer.
-b	Reports buffer activity.	<i>bread/s, bwrit/s</i> Transfers per second of data between system buffers and disk or other block devices. <i>lread/s, lwrit/s</i> Accesses of system buffers. <i>%rcache, %wcache</i> Cache hit ratios, (1-bread/lread) as a percentage. <i>pread/s, pwrit/s</i> Transfers via raw (physical) device mechanism.
-c	Reports system calls.	<i>scall/s</i> System calls of all types. <i>sread/s, swrit/s, fork/s, exec/s</i> Specific system calls. <i>rchar/s, wchar/s</i> Characters transferred by read and write system calls.

continued

Table 5-3 *continued*

Option	Description	Report Column Headings
-d	Reports activity for each block device, (disk or tape drive). When data displays, the device specification dsk- generally represents a disk drive. The device specification representing a tape drive is machine dependent.	<p><i>%busy, avque</i> Portion of time device was busy servicing a transfer request, average number of requests outstanding during that time.</p> <p><i>r+w/s, blks/s</i> Number of data transfers from or to device, number of bytes transferred in 512-byte units.</p> <p><i>avwait, avserv</i> Average time, in milliseconds, that transfer requests wait idly on queue, and average time to be serviced (includes seek, rotational latency, and data transfer times).</p>
-F	Reports on floating point activity.	<p><i>prfp</i> Number of processes requiring floating point hardware.</p> <p><i>%prfp</i> Percentage of processes requiring floating point hardware.</p> <p><i>prfpem</i> Number of processes requiring floating point emulation.</p> <p><i>%prfpem</i> Percentage of processes requiring floating point emulation.</p>
-g	Reports serial I/O.	<p><i>ovsiohw/s</i> Overflows at sio hardware.</p> <p><i>ovsiodma/s</i> Overflows at sio dma cache.</p> <p><i>ovclist/s</i> Overflows of clists.</p>

continued

Table 5-3 *continued*

Option	Description	Report Column Headings
-h	Reports buffer statistics.	<p><i>mpbuf/s</i> Number of mp (scatter-gather) buffers allocated per second.</p> <p><i>ompb/s</i> Number of times the system ran out of mp buffers per second.</p> <p><i>mphbuf/s</i> Number of mp buffer headers allocated per second.</p> <p><i>omphbuf/s</i> Number of times system ran out of mp buffer headers per second.</p> <p><i>pbuf/s</i> Number of physio buffers per second.</p> <p><i>spbuf/s</i> Number of sleeps/s waiting for physio buffers per second.</p> <p><i>dmabuf/s</i> Number of dma transfer buffers allocated per second.</p> <p><i>sdmabuf/s</i> Number of sleeps/s waiting for dma transfer buffers per second.</p>
-l	Reports on inter-system processor activity.	<p><i>cpuint_snd/s</i> Inter-system processor interrupts sent per second.</p> <p><i>cpuint_rcv/s</i> Inter-system processor interrupts received per second.</p> <p><i>IOcpuints/s</i> Inter-system processor interrupts per second for I/O.</p>
-m	Reports message and semaphore activities.	<p><i>msg/s, sema/s</i> Number of message and semaphore operations, per second.</p>

continued

Table 5-3 *continued*

Option	Description	Report Column Headings
-n	Reports name cache statistics.	<i>c_hits</i> , <i>cmisses</i> Number of name cache hits and misses. <i>hit%</i> The hit-to-miss ratio as a percentage.
-p	Reports paging activities.	<i>vflt/s</i> Address translation page faults (valid page not in memory). <i>pflt/s</i> Page faults from protection errors (illegal access to page) or "copy-on-writes". <i>pgfil/s</i> <i>vflt/s</i> satisfied by page-in from filesystem. <i>rclm/s</i> Valid pages reclaimed for free list.
-Q	Reports number of processes locked to processors. This produces a snapshot of activity at the end of the specified period.	<i>pltoCPU</i> Number of processes locked to processors. <i>%pltoCPU</i> Percentage of total number of processes running on the system locked to processors.
-q	Reports average queue length while occupied, and % of time occupied.	<i>runq-sz</i> , <i>%runocc</i> Run queue of processes in memory and runnable. <i>swpq-sz</i> , <i>%swpocc</i> Swap queue of processes swapped out but ready to run.
-R	Reports on process activity.	<i>dptch/s</i> Number of times dispatcher is run. <i>idler/s</i> Number of times idler is run per second. <i>swtdle/s</i> Number of times idler is switched to per second.

continued

Table 5-3 *continued*

Option	Description	Report Column Headings
-r	Reports unused memory pages and disk blocks.	<i>freemem</i> Average pages available to user processes. <i>freeswap</i> Disk blocks available for process swapping.
-u	Reports system processor use (the default).	<i>%usr, %sys, %wio, %idle</i> Portion of time running in user mode, running in system mode, idle with some process waiting for block I/O, and otherwise idle.
-v	Reports status of process, inode, file tables.	<i>proc-sz, inod-sz, file-sz, lock-sz</i> Entries/size for each table, evaluated once at sampling point. <i>ov</i> Overflows that occur between sampling points for each table.
-w	Reports system swapping and switching activity.	<i>swpin/s, swpot/s, bswin/s, bswot/s</i> Number of transfers and number of 512-byte units transferred for swapins and swapouts (including initial loading of some programs). <i>pswch/s</i> Process switches.
-y	Reports TTY device activity.	<i>rawch/s, canch/s, outch/s</i> Input character rate, input character rate processed by canon, output character rate. <i>rcvin/s, xmtin/s, madmin/s</i> Receive, transmit, and modem interrupt rates.

Exit Values

The **mpsar** command exits with a value 0 upon successful completion. If you specify an invalid option, **mpsar** exits with the value 2. It exits with value 1 for all other errors.

Diagnostic Messages

mpsar: Incompatible start and end times specified (*etime* <= *stime*)

The specified end time is the same as or before the start time.

mpsar: Time step and/or number of steps requested are invalid.

The time interval or number of intervals specified are not integer values, are negative values, or are otherwise invalid.

mpsar: Can't open filename.

The system cannot open the specified input file with the **-f** option.

mpsar: ofile same as ffile.

The specified input and output files are identical.

mpsar: argument -- illegal argument for option.

You specified an illegal argument for the option.

Command Examples

To see the current day's system processor activity:

```
mpsar
```

To watch system processor activity evolve for 10 minutes and save data:

```
mpsar -o temp 60 10
```

To review the disk and tape activity from that period at a later time:

```
mpsar -d -f temp
```

Warning

Running multiple copies of **sar**, **mpsar**, and **cpusar** can affect the results. The kernel automatically extracts the data using the three commands above. It is the extraction process not the collection that consumes resources. Therefore, results you produce when you run multiple copies might not reflect the actual performance of the system.

Files

/usr/bin/mpsar - command

/usr/adm/sa/sadd - daily data file

/usr/adm/sa/sar - daily report file

/usr/lib/sa/sa.adrfl - address file

Standards Conformance

The **mpsar** command is not part of any currently supported standard. It is an extension of AT&T System V provided by The Santa Cruz Operation, Inc.

The cpqmon Command

Purpose: Screen-Oriented System Monitor for SCO UNIX

Syntax: cpqmon [-Pp] [-[ntswidrM]] [-N NIC]

Description: **cpqmon** 1.0, a "performance" monitor for SCO UNIX. The **cpqmon** command uses curses(S)/terminfo(S) to display the system performance information, therefore any ASCII terminal works. Use the **cpqmon** command to display disk I/O, internal system counters, and memory usage statistics.

Basic Operation

The screen splits into two sections on a 25-line screen and three sections on a 43-line screen. The top section shows a histogram of system activity divided into two parts, system processor activity and "Waiting on I/O." The system processor use displays with smoothing of 1 ("instant" if the update interval is 1 second), 5, and 10 seconds.

The screen shows total system processor use with user, kernel, and "break" subdivided. Most performance utilities (**vmstat**) lump kernel CPU_KERNEL) and wait (CPU_WAIT) times together as kernel time. **cpqmon** considers CPU_WAIT time as idle.

On a color monitor, total system processor use displays in green if the system processor use is below 70 percent, yellow if it is between 70 and 89 percent, and red if it is 90 percent or above. The "Wait" display shows the 1-, 5-, and 10-second smoothed percentages of real time when no process could be run because otherwise ready-to-run processes were waiting on logical, swap, or physical I/O. On a color monitor, total wait time displays in green if it is below 30 percent, yellow if it is between 30 and 49 percent, and red if it is 50 percent or above.

The second part of the screen displays the Sysinfo/Minfo or "main" screen. On a 43-line display, extra information displays on the bottom of the screen. By typing the screen command, the second screen on a 24-line display or the third screen on a 43-line display changes to reflect that screen's information.

On a 43-line display, the "main" screen always remains on the second section of the display. On a color monitor, static numeric values such as "maxmem" display in blue (the same color as screen literals/labels). Dynamic numeric values display in green, with the exception of total system processor and wait percentages, which display in light green, red, or yellow.

An "INEXACT" indicator on the top line means that **cpqmon** was not scheduled quickly enough to capture accurate 1-second (nominal) values. Continued INEXACT indicators suggest the 5- and 10-second smoothed values are also wrong. An "INVALID" indicator means **cpqmon** was scheduled three or more seconds late, and all percentage displays are now suspect.

Commands

- +/-** These commands add or subtract one second from the update interval. The value at startup is 2 seconds, and the range is 1 to 4. You can determine the update interval by looking at the top system processor histogram, labeled "Instant" for a 1-second interval, and "X sec" for 2-4 second intervals.
- ESC/q** Exit **cpqmon**.
- l** Lock **cpqmon** into memory. If you do this, PLOCK appears at the top of the screen. This indicates you are using excessive amounts of system resources. The **cpqmon** process is not listed in a process status display because SSYS (locked, resident) processes are not shown.
- u** Unlock **cpqmon** from memory.
- Space bar** Toggle system processors (MPX Only). Pressing the space bar displays information from all system processors to each installed system processor. This command only works with SCO UNIX 3.2v4.2 or later.

Screen Commands

- m** Display Main screen. This screen provides information taken from the sysinfo and minfo structures. The sysinfo structure contains information on the system processor use, system calls, process information, and IPC. The minfo structure contains information on the memory subsystem.
- e** Display Extra screen. This screen provides information on the settings of some system tunables as well as boot and process information. Standard ANSI terminals truncate the boot information, due to screen limitations.
- p/P** Display Process screen. This displays a process status, overlaying Sysinfo/Minfo on a 25-line screen or Var/Bootinfo/Tune/Proc on a 43-line screen. On a 43-line screen, **P** displays a larger ps display, overlaying Sysinfo/Minfo and Var/Bootinfo/Tune/Proc.

The process information consists of the following 11 columns:

1 Two-character status.

1st Character - process status:

s - sleeping

R - ready to run

z - zombie

d - stopped by the debugger

i - idle

p - running on processor (on single system processor systems, only **cpqmon** shows this)

x - XBREAK - process is growing or shrinking

2nd Character - process swap status:

S - process is swapped

- blank - process is in memory

2 User name of the owner of the process. If the process is running with setuid, a '#' appears to the right of the user name.

3Process ID.

4Measure of recent system processor use by the process.

5Process Priority.

6Process nice value.

7System processor User Time used by the process.

8System processor System Time used by the process.

9 Size (in kilobytes) of the process virtual data and stack segments.

10Terminal that this process is originating from.

11Process command.

On color systems, ready-to-run processes display in yellow, unless they are ready but swapped out. In that case, they display in red.

Because a limited space is available for displaying process status, particularly on a 25-line screen, a selective elimination algorithm reduces the list when insufficient room is available.

init (pid 1) and system/resident (SSYS) processes never display. When a display cycle begins and there is not room to display all of the processes, processes are eliminated in the following order:

1. *getty*, *uugetty*, *sh*, *cs*, and *ksh*
2. swapped and zombie processes
3. no system processor processes (no system processor time during last cycle)
4. sleeping processes

If there still is insufficient room, a message to that effect displays.

s Display serial I/O screen.

- n** Display Streams Data. This display dynamically shows STREAMS queues and buffers use.
 - M** Display MAC Layer Stats. This display shows the Media Access Control (MAC) statistics. The lowest level of networking software in the system maintains these statistics.
 - t** Display Table Stats. The current and maximum occupancy of various kernel tables display. This contains tables such as the File table, Inode table, Proc table, and so on.
 - w** Disk (Winchester) display. This display shows each disk and diskette device known to the system and performance statistics for each. The percentages are inaccurate for one or two display cycles after the 'w' (Winchester) selection becomes active. Further updates might also be inaccurate due to either noisy kernel data or data capture latency.
 - i** Compaq IDA (Intelligent Drive Array) display. This display shows each IDA controller and logical volume known to the system and performance statistics for each.
-

- d** Disk (SCSI) display. This display shows each SCSI disk that uses the Sdsk interface and performance statistics for each.
- r** Removable Media display. This display shows each SCSI tape, SCSI CDROM, ct tape, and the performance statistics for each.

Command Line Options

cpqmon accepts the following command line options:

Table 5-4
cpqmon Command Line Options

Option	Description
-P	Invoke cpqmon with long ps display (if 43-line screen).
-p	Invoke cpqmon with short ps display.
-n	Invoke with network display.
-t	Invoke with table display.
-s	Invoke with sio display.
-w	Invoke with disk (Winchester) display.
-i	Invoke with IDA display.
-d	Invoke with Sdsk display.
-r	Invoke with removable media display.
-M	Invoke with MAC layer display.
-N NIC	Specifies the device name of the NIC to use. If you do not specify an option, cpqmon uses cet0 as the default network card.

Screen Information

The Sysinfo/Minfo screens consist of information from the sysinfo and minfo structures in the kernel. The following table lists the screen fields and a short description for each field.

**Table 5-5
Sysinfo/Minfo Screen Fields**

Field	Description
bread	Actual buffers (disk or tape blocks) read.
bwrite	Actual buffers (disk or tape blocks) written.
lread	Logical buffers (disk or tape blocks) read.
lwrite	Logical buffers (disk or tape blocks) written.
phread	Physical reads.
phwrite	Physical writes.
swapin	Logical blocks swapped in.
swapout	Logical blocks swapped out.
bswapin	Actual blocks swapped in.
bswapout	Actual blocks swapped in.
iget	Get inode.
namei	Lookup inode for pathname.

continued

Table 5-5 *continued*

Field	Description
dirblk	Directory blocks read.
readch	Characters read by read() system call.
writch	Characters written by write() system call.
rawch	Raw tty character read.
canch	Raw tty character put onto canonical queue.
outch	tty characters written.
msg	Message operations (msgsnd()).
sema	Semaphore operations (semop()).
maxmem	Maximum amount of user memory.
frmem	Available amount of user memory.
mem used (%)	Percentage of maximum user memory available.
nswap	Maximum amount of swap space.
frswp	Available amount of swap space.
swp used (%)	Percentage of maximum swap space available.
pswitch	Process context switches.
syscall	System calls.
sysread	Read() calls.
syswrit	Write() calls.

continued

Table 5-5 *continued*

Field	Description
sysfork	Fork() calls.
sysexec	exec[,l,e,etc.]() calls.
runque	Number of processes placed on run queue.
runocc	Number of processes -now- on run queue.
swpque	Number of processes placed on swap queue.
swpocc	Number of processes -now- on swap queue.
rmem	Available resident (not swapable) memory.
smem	Available swapable memory.
vfault	Translation fault.
demand	Demand zero and demand fill pages.
pfault	Protection fault.
steal	Number of pages stolen.
frdpgs	Pages freed.
vfpg	Pages are freed by vhand.
sfpg	Pages are freed by sched.
vspg	Pages are freed/swapped by vhand.
sspg	Pages are freed/swapped by sched.
pnpfault	Page not present fault.
wrtfault	Write fault.

continued

Table 5-5 *continued*

Field	Description
unmodsw	Unmodified pages on swap.
unmodfl	Unmodified pages in file.
psoutok	i386 swapping out a process.
psinfai	i386 swapping in a process failed.
psinok	i386 swapping in a process succeeded.
rsout	i386 swapping out a region.
rsin	i386 swapping in a region.
swap	Pages on swap.
cache	Pages on cache.
file	Pages on file.

**Table 5-6
Var Screen Fields**

Field	Description
v_autoup	NAUTOUP age (in seconds) before BDFLUSH writes a delayed-write buffer.
v_buf NBUF	Number of I/O buffers.
v_clist NCLIST	Number of clists allocated.
v_file NFILE	Size of file table.
v_hbuf NHBUF	Hash buffers (power of 2).
v_inode NINODE	Size of incore inode table.
v_maxpmem MAXPMEM	Maximum physical memory to use.
v_maxup MAXUP	Maximum processes per user.
v_mount NMOUNT	Size of mount table.
v_pbuf NPBUF	Number of physical I/O buffers.
v_proc NPROC	Size of proc table.
v_region NREGION	Number of regions allocated.
v_vhndfrac VHNDFRAC	Fraction of maxmem limits vhand.

Table 5-7
Tune Screen Fields

Field	Description
t_ageintvl	Age process after so many seconds.
t_bdflushr	The rate at which BDFLUSHR is run in seconds.
t_gpgshi	Once system starts to steal pages, do not stop until freemem > t_getpgshi.
t_gpgslo	If freemem < t_getpgslo, then start to steal pages from processes.
t_gpgsmask	Mask used by getpages to determine if a page is stealable.
t_maxfc	Maximum number of pages that will be saved up and freed at once.
t_maxsc	Maximum number of pages that will be swapped out on a single operation.
t_maxumem	Maximum size of a user's virtual address space, in pages.
t_minarmem	Minimum available resident memory to maintain to avoid deadlock in pages.
t_minasmem	Minimum available swappable memory to maintain to avoid deadlock in pages.

Table 5-8
Proc Screen Fields

Field	Description
sleep	Awaiting an event.
run	Running.
zombie	Process terminated but not waited for.
stop	Process stopped by debugger.
idle	Intermediate state in process creation.
onproc	Process is being run on a processor.
xbrk	Process being xswapped.
total	All processes.
in mem	All processes except those swapped out.

Notes

IDA Statistics are only available with SCO UNIX version 3.2v4.2 and later.

Authors

cpqmon was derived from u386mon written by Warren Tucker. Enhancements were made by Compaq Computer Corporation.

The SYSDEF Command

Purpose: Display current tunable parameter settings.

Syntax: sysdef

Description: Prints the values currently set for all tunable kernel parameters. This command generates the output by analyzing the named operating system file (*/unix* by default) and extracting the configuration information from the name list.

The UPTIME Command

Purpose: Report current time, system uptime, and load average.

Syntax: uptime

Description: The **uptime** command reports the current time of day, how long the system has been up, the number of users logged on the system, and the system load average over the last minute, the last five minutes, and the last 15 minutes. The load average is the average number of processes in the run queue of the kernel during an interval.

The load average number is a good indicator of system processor loading. Use this value, together with system parameters from the **sar** and **ps** utilities, to identify system processor bottlenecks.

Index

\$

\$PATH environment variable 4-11

A

Acer Fast File System (AFS) 3-28

Active processes, monitoring 4-9

AGEINTERVAL, tuning 2-22

B

Back ups, to raw disk partition 3-5

Block I/O

monitoring with sar command
3-9

system call activities, monitoring
3-11

Boot messages 2-3

Bottlenecks

disk subsystem

identifying 3-4

saturating the COMPAQ

SMART SCSI Array

Controller 3-3

system performance factors

in applications using raw
devices 3-2

system performance factors

in UNIX filesystems 3-
2

tuning guidelines 3-1

using sar -u to identify

potential disk subsystem
bottleneck 3-5

Bottlenecks *continued*

disk subsystem, indications 3-10

memory subsystem

balancing workloads 2-21

causes of 2-1

identifying 2-3

swap and paging activities 2-5

tuning guidelines 2-16

tuning kernel parameters 2-21

system processor

identifying 4-2

tuning guidelines 4-10

using mpstat to monitor

system processor use 4-3

using ps -el and -ef to

identify processes 4-9

using sar -uq to identify

potential bottleneck 4-4

Buffer block I/O, monitoring with sar

command 3-9

Buffer cache

block I/O 3-4

guideline for determining size 2-4

NBUF parameter 2-17

tuning 3-22

C

cache hit, defined 3-1

cache miss, defined 3-1

Command summary

sysdef 5-35

uptime 5-35

COMPAQ EISA Configuration

Utility 3-21

- COMPAQ Intelligent Drive Array
 - Controller 3-1
- COMPAQ Intelligent Drive Array
 - Controller-2 3-1
- COMPAQ SMART SCSI Array
 - Controller
 - adding an additional controller 3-19
 - determining activity level 3-10
 - features and capabilities 3-3
 - limitations of TechNote 3-1
 - lowering thresholds for drive mirroring and data guarding 3-10
 - write-posting cache and raw disk I/O 3-5
- COMPAQ supplement for SCO UNIX 1-2
- Corollary Multiprocessor Activity
 - Monitor (mpstat), monitoring system processor use 4-3
- cpio command, backing up and restoring files 3-26
- cpqmon
 - detailed information 5-22
 - types of information displayed 1-2
 - where to find it 1-2
- cron program
 - memory bottlenecks and workloads 2-21
 - overview 1-4
- crontab 2-21, 4-10

D

- Data guarding
 - effect on disk storage capacity 3-22
 - explained 3-17
- Data striping, explained 3-16
- Databases, accessing data in raw disk partitions 3-5
- Device driver, factor affecting disk subsystem performance 3-2
- Directories
 - \$PATH environment variable 4-11
 - inodes in, counting 3-26
- Disk subsystem
 - Acer Fast File System (AFS) 3-28
 - bottlenecks, identifying 3-4
 - COMPAQ product limitation note 3-1
 - creating logical drives 3-21
 - databases, buffer cache 3-5
 - directories, counting inodes 3-26
 - fault tolerance, limiting 3-22
 - filesystems, maintaining efficient 3-24
 - paging and I/O 3-4
 - raw device I/O 3-2
 - reducing fragmentation 3-25
 - tuning guidelines 3-15
 - UNIX filesystem I/O 3-1
- Distributed parity, explained 3-18
- divvy command 3-21
- Drive mirroring
 - disk performance 3-22
 - explained 3-16

E

etc/swap command 2-6

F

Fault tolerance

limiting for optimal performance
3-22

RAID 0 3-16

RAID 1 3-16

RAID 4 3-17

RAID 5 3-18

fdisk command 3-21

Filesystems

directories, recommendation for
keeping small 3-26

disk fragmentation, reducing 3-25

distributing for optimal disk
performance 3-19

free space, maintaining 3-24

maintaining efficient 3-24

name-cache parameters, tuning
3-27

scatter-gather I/O parameters,
tuning 3-28

small vs. large 3-24

G

GPGSHI

relationship to available free
memory 2-1

tuning 2-22

using sysdef to determine value
2-6

GPGSLO

relationship to available free
memory 2-1

tuning 2-22

using sysdef to determine value
2-6

H

Hangs, caused by full filesystem 3-24

I

I/O buffers, determining amount of
memory used 2-4

Illustration

drive mirroring 3-16

K

Kernel

memory requirements summary 2-
1

parameters

AGEINTERVAL 2-22

amount of system memory
gained from tuning 2-16

buffer cache 2-23

displaying values with
sysdef 1-5

GPGSHI 2-1, 2-22

GPGSLO 2-1, 2-22

NBUF 2-17, 2-23

NHBUF 2-23

rebuilding with sysadmsh
2-3

Kernel *continued*
recommended tuning approach
2-21
tuning 2-21
using boot messages to help
tune 2-3

L

Load average, monitoring 4-3
Logical drives, creating 3-20
Logical I/O 3-4

M

MAXUP, tuning 2-25
Memory subsystem
adding more memory 2-16
available free memory, effect on
system performance 2-1
bottlenecks
balancing workloads 2-21
causes of 2-1
identifying with sar-w option
report 2-7
monitoring user processes 2-4
swap and paging activities 2-5
critical shortage, effect on
system performance 2-2
determining allocation 2-3
finding amount of free memory
2-6
kernel parameters and tuning 2-21
recommended swap space 2-2
shortage and swapping activities
2-7
tuning 2-1
tuning guidelines 2-16
mpsar command, detailed command
information 5-14
mpstat command, monitoring system
processor use 4-4

N

Name-cache parameters, tuning 3-27

NBUF 2-23
NCALL, tuning 2-25
NCLIST, tuning 2-25
NFILE, tuning 2-25
nice command 4-11
NINODE, tuning 2-25
NMOUNT, tuning 2-25
NMPBUF, tuning 3-28
NMPHEADBUF, tuning 3-28
NPROC, tuning 2-25
NREGION, tuning 2-25

O

Occupancy rate of small filesystems,
maximum 3-24
Option reports, See sar command 3-5

P

Page-fault handler 2-2, 2-8
Page-stealing daemon 2-2
Paging, investigating with sar
command 2-8

- Parameters
 - kernel 2-21
 - kernel buffer cache 3-22
 - name-cache 3-27
 - scatter-gather 3-28
 - Parameters, *See also* Buffer cache parameters, Kernel tables parameters, Paging parameters, STREAMS parameters, 2-21
- Parity drive 3-17
- Performance
 - commands summary 5-1
 - disk subsystem 3-1
 - memory subsystem 2-1
 - system processor 4-1
- Performance analysis tools
 - COMPAQ cpqmon 1-2, 5-22
 - COMPAQ Server Manager/R 1-2
 - cron daemon 1-4
 - definition 1-2
 - mpsar command 5-14
 - ps command 5-1
 - sadc program 1-4, 5-11
 - sar command 1-4, 5-4
 - sysdef 5-35
 - UNIX tools compared to third-party tools 1-2
 - uptime command 5-35
- Physical block I/O 3-4
- Process state, capturing with ps command 1-3
- Process working set, defined 2-1
- ps command
 - detailed command information 5-1
 - determining memory usage information 2-21
 - el options report columns explained 2-4, 4-9
 - invoking through a script 1-3
 - monitoring user processes 2-4
 - overview 1-3
 - suggested use with sar command 1-3, 2-21
 - summary of memory information reported 2-5
 - using -el options to determine memory usage information 2-16

R

- RAID 0, *See* Data striping 3-10
- RAID 1, *See* Drive mirroring 3-10
- RAID 4, *See* Data guarding 3-10
- RAID 5, *See* Distributed parity 3-10
- Raw block I/O, monitoring 3-9
- Raw devices
 - explained 3-2
 - I/O and buffer cache 3-5
 - optimal disk performance 3-19
- Reference commands 5-1

S

S5CACHEENTS, tuning 3-27

S5HASHQS, tuning 3-27

S5OFBIAS, tuning 3-27

sadc program

overview 1-4

sampling system counter
information 5-11

storage of binary system activity
data 1-4

sar command

-b and -c options and controller
activity level 3-9

comparing kernel tables
parameters with -v option 2-26

detailed command information
5-4

identifying disk subsystem
bottlenecks 3-4

identifying memory subsystem
bottlenecks 2-21

investigating paging activities
2-8

monitoring load average 4-3

monitoring system processor use
4-5

monitoring system processor
wait time 3-5

option report description

-b option 3-9

-c option 3-11

-p option 2-8

-q option 4-5

-r option 2-6

-u option 3-5, 4-4

-w option 2-7

sar command *continued*

recommended monitoring

interval 1-5, 2-5

running without time interval
option 1-4

suggested use with sadc 1-4

use for system snapshots 1-5

SCO UNIX System V/386 Operating
System, I/O 3-1

size command 2-5, 2-16

Snapshot 1-5, 2-4

Swap and paging activities,
monitoring 2-5

swap command, -l option report
columns explained 2-6

Swap space

/etc/swap 2-6

recommended size 2-2

sysadmsh

backing up files 3-25

displaying kernel parameters 1-5

tuning kernel parameters
2-3, 3-22, 4-12

sysdef command

displaying kernel parameters
1-5, 4-12

finding values of GPGSLO and
GPGSHI 2-6

reference summary 5-35

System activities

balancing system workload 2-21

capturing information with sar 1-4

recommended monitoring interval,

for system activities 1-5

snapshots 1-5

System activities *continued*

- swap and paging 2-5, 3-4
- what to monitor to identify disk
 - subsystem bottleneck 3-4
- what to monitor to identify memory subsystem
 - bottleneck 2-3
- what to monitor to identify system processor bottleneck 4-2

System administrator, using ps command 1-3

System processes

- capturing information 1-3
- using the ps command 1-3

System processor

- board, adding 4-10
- bottlenecks, identifying with
 - sar -uq options 4-4
- improving efficiency of
 - user-defined features 4-11
- list of activity types 4-1
- performance measurements
 - for load average 4-3
 - monitoring system processor use 4-3
- system processor use, defined 4-1
- tuning 4-1, 4-10
- workload balancing 4-10

T

TechNote

- intended document audience 1-1
- issues addressed 1-1
- performance factors beyond
 - scope of TechNote 1-1
- use for product evaluation 1-2

Tuning

- disk subsystem 3-1, 3-15
- kernel parameters 4-12
- memory subsystem 2-1, 2-16
- system processor 4-1, 4-10

U

UNIX

- cron program 1-4, 2-21
- divvy command 3-21
- etc/swap command 2-6
- fdisk command 3-21
- filesystem data access described 3-1
- nice command 4-11
- ps command
 - el options report columns explained 2-5, 5-2
 - monitoring user processes with -el option 2-4
 - overview 1-3
- sadc program 1-4
- sar command 1-4, 2-21
- sar -v option to determine kernel tables parameters 2-26
- size command 2-5, 2-16

UNIX *continued*

sysadmsh

1-5, 2-3, 3-22, 3-25, 4-12

sysdef command

1-5, 2-6, 4-12, 5-35

uptime command 4-3, 5-35

uptime command

monitoring load average 4-3

reference summary 5-35
