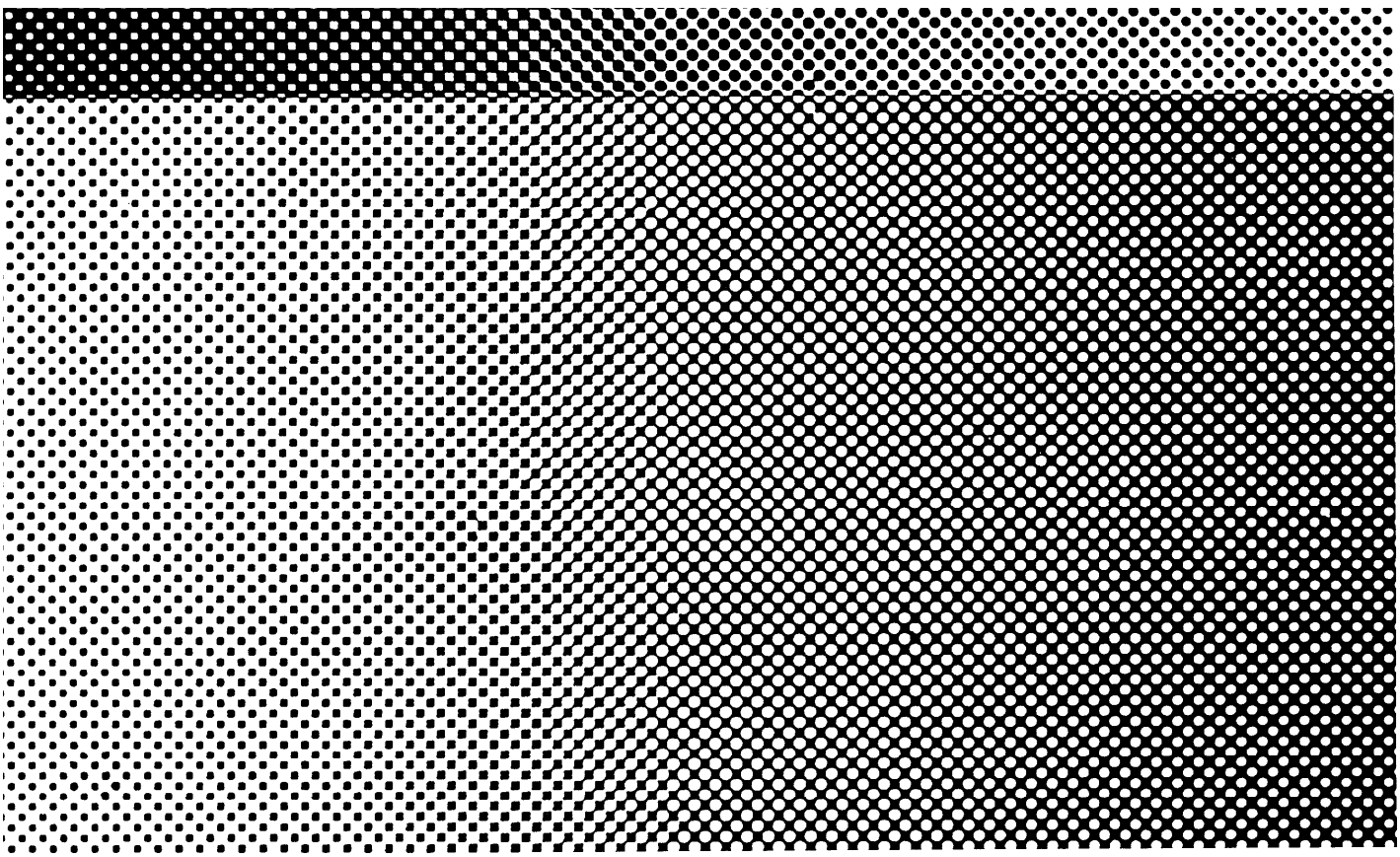
305-607
Issue 2

AT&T 3B2 Computer

System Performance Analysis

Utilities Guide



TRADEMARKS

The following trademark is used in this manual:

- UNIX — Registered trademark of AT&T

ORDERING INFORMATION

Additional copies of this document can be ordered by calling

Toll free: 1-800-432-6600 In the U.S.A.

1-800-255-1242 In Canada

Toll: 1-317-352-8557 Worldwide

OR by writing to:

AT&T Customer Information Center
Attn: Customer Service Representative
P.O. Box 19901
Indianapolis, IN 46219

NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

Table of Contents

List of Figures

Chapter 1: Introduction

General

The System Performance Analysis Utilities provide system administrators and software developers with commands for collecting and examining system usage data. System usage data can be used to analyze the present performance of the computer and determine load balancing and system tuning strategies that will improve performance.

This document contains a high-level description of performance management. The strategy that you use to improve the performance of your computer is dependent on the UNIX* Operating System, application programs, and hardware installed. A more detailed description of performance management is contained in the *AT&T 3B2 Computer, UNIX System V, System Administrator's Guide.*

This document describes each command in the System Performance Analysis Utilities. The descriptions include the purpose of the command, how to use the options, and examples of how to use the command.

* UNIX is a registered trademark of AT&T

Installing System Performance Analysis Utilities

There are two floppy diskettes in your System Performance Analysis Utilities package that are labeled as follows:

System Performance Analysis Utilities - Release 1.1 (For use with SVR2.0.5)

System Performance Analysis Utilities - Release 1.1 (For use with SVR3)

The floppy that you use to install the utilities depends on the version of the UNIX Operating System that your computer is running. The labels identify the release that the floppy is to be used with. (SVR3 implies Release 3.0 or a later release.)

It is important that you install the utilities from the right floppy. If you are not sure what release your computer is running, enter the command **uname -r** to display the system release.

Before you install the appropriate System Performance Analysis Utilities, you must remove any release of Performance Measurement Utilities that may have been previously installed. For the procedure to install and remove software, refer to your *AT&T 3B2 Computer, UNIX System V, Owner/Operator Manual.*

Check the amount of free space in the **root** and **/usr** directories before installation. This package requires 84 blocks in **root** and 566 blocks in **/usr.**

Summary of System Performance Analysis Commands

The System Performance Analysis Utilities contain ten UNIX System commands and two shell scripts that are listed below:

prfdc	Performs the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed.
prfld	Used to initialize the recording mechanism in the system.
prfpr	Formats the data collected by **prfdc** or **prfsnap.**
prfsnap	Collects data (like **prfdc)** at the time of invocation only.
prfstat	Used to enable, disable, or check the status of the sampling mechanism.
sadc	Used to sample and save the system activity data.
sadp	Reports disk access location and seek distance in tabular or histogram form.
sag	Graphically displays the system activity data stored in a binary data file by a previous **sar** command.
sar	Calls **sadc** to sample cumulative activity counters in the operating system at specified intervals of time and will save the samples in binary format.
sa1	Shell script used to collect and store data in binary file **/usr/adm/sa/sa***dd* where *dd* is the current day.
sa2	Shell script that writes a daily report in file **/usr/adm/sa/sar** *dd*.
timex	When **timex** and another **command** are executed, the elapsed time, user time, and system time spent in execution of **command** are reported in seconds.

To use any of the commands beginning with **prf,** you must be logged in as **root.** You must also be logged in as **root** to use **sa1** and **sa2.** All the other commands are available to all users.

Conventions Used to Describe Commands

When command formats are described, the following symbology and conventions are used to define the command syntax.

- The basic command is shown in bold type. For example: **command** is in bold type.
- Arguments that you must supply to the command are shown in italics. For example: **command** *argument*
- Command options and arguments that do not have to be supplied are enclosed in brackets ([]). For example: **command** *[optional arguments]*
- The pipe symbol "¦" is used to separate arguments when one of several forms of an argument can be used for a given argument field. The pipe symbol can be thought of as an exclusive OR function in this context. For example: **command** *[argument1 ¦ argument2]*

In the sample command discussions, the command lines that you input are ended with a carriage return. This is shown by using <*CR*> at the end of the command lines.

Screen Display Conventions

The following conventions are used in this manual to show terminal inputs and system responses.

This style of type is used to show system generated responses displayed on your screen.

This style of bold type is used to show inputs entered from your keyboard that are displayed on your screen.

These bracket symbols, < >, identify inputs from the keyboard that are not displayed on your screen, such as: <CR> carriage return, <CTRL d> control d, <ESC g> escape g, passwords, and tabs.

This style of italic type is used for notes that provide you with additional information. This style of type is also used to show variable inputs such as times and dates.

Screen displays are included in this document to help you understand described procedures. These are only examples of the system responses that you should receive. System responses that you receive may differ for various reasons. For example, some system generated responses are affected by the utilities that are installed. Also, it is unlikely that variables, such as file sizes, times, dates, etc. used in the examples will agree with the displays on your terminal screen.

Manual Organization

Chapter 1, "Introduction," describes the purpose and content of the document, installation of the utilities, commands, conventions used for commands and displays, and manual organization.

Chapter 2, "Performance Management," describes basic system tuning and load balancing strategies. This description covers finding problems, fixing problems, and improving performance.

The commands in the System Performance Analysis Utilities fall into one of three categories: kernel profiling, system activity report package, or performance tools. Each of the following chapters in this document describes the commands in one of these categories.

Chapter 3, "Kernel Profiling," describes the commands that are used to determine the routines the operating system is spending its time on during operation. The commands included in kernel profiling are: **prfld, prfstat, prfdc, prfsnap,** and **prfpr.**

Chapter 4, "System Activity Report Package," describes the commands that are used to sample, save, and process system activity data. The command and shell scripts included in this category are: **sadc, sa1,** and **sa2.**

Chapter 5, "Performance Tools," describes the commands that are used to display the performance of the operating system. The commands that make up the performance tools are: **sar, sadp, sag,** and **timex.**

Chapter 2: Performance Management

General

Performance management involves tuning your 3B2 Computer for optimum efficiency. The followng is a brief description of performance management. A detailed description is contained in the *AT&T 3B2 Computer, UNIX System V, System Administrator's Guide.*

When you boot your computer, tunable configuration parameters are automatically set to a basic configuration. This configuration is satisfactory for most applications. However, the basic configuration cannot take into account the usage patterns and the behavior of every possible application. Therefore, the basic configuration may not be the best for your particular applications. For this reason, the structure of the system allows you to reconfigure it to enhance the performance for your applications.

Finding Problems

There are a number of indicators that can be used to determine if your computer needs to be tuned. For example, you may see that the response time at the console is frequently slow or a particular job takes longer to run.

Hardware and software changes can affect the efficiency of a computer. If there is a noticeable decrease in performance after adding or removing hardware or software, reconfiguration of the system may be necessary. Disk utilization can also affect the efficiency of a computer. If there is a noticeable decrease in performance due to disk Input/Output (I/O), the file system may need to be reorganized.

Fixing Problems

Data collected using performance tools will indicate what corrective action needs to be taken, see Chapter 5, "Performance Tools." Some of the major areas for action are:

- Modifying the tunable configuration parameters.
- Uninstalling optional kernel packages not needed by your applications.
- Improving disk utilization.
- Defining best system usage patterns.

These areas are covered in detail in the *System Administrator's Guide.*

Improving Performance

Modifying the Tunable Configuration Parameters

Generally, the default parameters will result in acceptable performance. If, however, you are running an application that has special performance needs, you can use the tools described in Chapter 5, "Performance Tools," to measure system load and determine which parameters might be changed to improve performance. The *System Administrator's Guide* contains detailed information on all tunable parameters.

Improving Disk Utilization

A disk I/O bottleneck can decrease system performance. Some ways to improve the performance of the disk subsystem are:

- Allocate a larger buffer cache.
- Set the text bit (sticky-bit) for frequently used executables.
- Organize the file systems to minimize disk activity.
- Set the logical block size to suit the application.

These items are covered in detail in the *System Administrator's Guide.*

Chapter 3: Kernel Profiling

General

Kernel profiling is a mechanism that allows you to determine where the operating system is spending its time during operation. It consists of a pseudo-device, **/dev/prf,** the associated device driver, and user level commands to control the profiling process and to generate reports. The profiling mechanism samples the program counter on every clock interrupt and increments the counter corresponding to the function shown by that value of the program counter.

The user level kernel profiling commands are **prfld, prfstat, prfdc, prfsnap,** and **prfpr.** These commands are described on the next several pages.

prfld Command

The **prfld** command initializes, or loads, the recording mechanism in the system. The **prfld** command has the following format:

/etc/prfld *[namelist]*

The **prfld** command generates a table, in memory, containing the starting address of each subroutine as extracted from *namelist.* The default of *namelist* is **/unix.**

The following command generates a table in memory containing the starting addresses of each subroutine:

/etc/prfld <*CR*>

prfstat Command

The **prfstat** command enables or disables the sampling mechanism initialized by **prfld.** The **prfstat** command has the following format:

/etc/prfstat *[on ¦ off]*

Profiler overhead is less than 1 percent as calculated for 500 text addresses.

Enter **prfstat** without an argument to find out the status of the profiler.

```
#/etc/prfstat <CR>
profiling (enable or disabled)
2278 kernel text addresses
#
```

Enter **prfstat** with the argument *on* to enable or turn on the recording mechanism.

```
#/etc/prfstat on<CR>
profiling enabled
2278 kernel text addresses
#
```

Enter **prfstat** with the argument *off* to disable or turn off the recording mechanism.

```
#/etc/prfstat off<CR>
profiling disabled
2278 kernel text addresses
#
```

prfdc Command

The **prfdc** command performs the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. The **prfdc** command has the following format:

/etc/prfdc *file [period [off hour]]*

The **prfdc** command stores the contents of the counters in a *file* every *period* minutes and turns off at *off hour*. Valid values for *off hour* are 0 through 24.

The following command copies the current value of all the text address counters into a file called *temp* every five minutes and turns off at 4:00 p.m.:

/etc/prfdc temp 5 16<*CR*>

prfsnap Command

The **prfsnap** command performs the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. The **prfsnap** command has the following format:

/etc/prfsnap *file*

The **prfdc** command stores the counters in *file* every specified amount of minutes and turns off at a specified hour. The **prfsnap** command, however, takes a "snapshot" of the system, collecting data at the time it is called, and appends the counter values to *file.*

The following command copies the current value of all the text address counters into a file called *temp*, at the time the command is entered:

/etc/prfsnap temp <*CR*>

prfpr Command

The **prfpr** command formats the contents of *file* (data that was collected by **prfdc** or **prfsnap).** The **prfpr** command has the following format:

/etc/prfpr *file [cutoff [namelist]]*

Each text address is converted to the nearest text symbol (function) and the percentage of time used by that function is printed if the activity percentage is greater than the cutoff number that you specify. The range of *cutoff* is 0 percent to 99 percent where 0 percent prints all contents. The default of *cutoff* is 1 percent; the default of *namelist* is **/unix.**

The screen display on the next page illustrates the use of **prfpr** to display data collected by **prfdc** or **prfsnap.**

```
#/etc/prfpr temp 0<CR>

04/01/88 09:06
04/01/88 09:06

idle      93.0
fbclop    0.5
svirtophys 0.1
reglock   0.1
findreg   0.1
wakeup    0.1
pswtch    0.1
systrap   0.1
bread     0.1
getblk    0.1
bflush    0.5
bdflush   0.3
ttxput    0.1
putc      0.3
intoB     0.1
s5writei  0.1
s5bmap    0.1
idstrategy 0.1
idsetup   0.1
idseek    0.1
idxfer    0.1
idint     0.2
user     h0.7
#
```

Note: These are the function calls in the kernel. For detailed information on function calls, refer to the AT&T 3B2 Computer Programmer Reference Manual. See sections on Subroutines, System Calls, and/or Library.

How to Operate the System Profiler

The system profiler (prf) initializes the recording mechanism. It will generate a table containing the starting address of each system subroutine as extracted from the UNIX System kernel. The requirements for operating the system profiler are defined as follows:

Step 1: Enter **/etc/prfld** to initialize or load the profiler.

> **Note:** The maximum number of text symbols that the profiler can look at is specified by the tunable parameter, PRFMAX. The default value of PRFMAX is 2048. If you exceed this value, an error message
>
> ```
> too many text symbols
> ```
>
> is displayed when you try to execute **etc/prfld.** To increase the number of allowable text symbols, edit the file **/etc/master.d/prf,** change the value to a larger number, and rebuild the operating system. (See the procedure for reconfiguring the system in the *System Administrator's Guide.)*

Step 2: Enter **/etc/prfstat on** to turn on the sampling mechanism.

Step 3: Enter **/etc/prfdc** *filename period off hour* to collect and enter the data into a file. *Period* specifies the number of minutes to run the profiler. *Off hour* is the time (hour) you want to stop the profiler (0 through 24).

> **Note:** A snapshot of the data can also be taken by using the **prfsnap** command.

Step 4: Enter **/etc/prfpr** *file* to print the contents of the data, collected by either **prfdc** or **prfsnap.**

Step 5: Enter **/etc/prfstat off** to turn off the sampling mechanism.

Step 6: Enter **/etc/prfstat** at any time to check the status of the sampling mechanism.

The system profiler must be loaded and turned on after every boot. If you want the profiler to begin automatically when you boot the system, you can create a file (shell script) in **/etc/rc.d/** called **prf.** The contents of this file should be as follows:

```
#Load the profiler and enable operation.
/etc/prfld
/etc/prfstat  on
```

The **prf** shell script will be executed during system initialization and the following messages will be displayed:

```
profiling enabled

xxx kernel text addresses

Note: Where xxx states how many kernel text addresses
are in the present UNIX System kernel.
```

Chapter 4: System Activity Report Package

General

As various system actions occur, counters in the operating system are incremented to keep track of these activities. System activities that are tracked are listed below:

- Central Processing Unit (CPU) utilization
- Buffer usage
- Disk and tape input/output activity
- Terminal device activity
- Switching and system-call activity
- File access
- Queue activity
- Inter-process communications
- Paging
- Remote File Sharing.

System activity data can be accessed on a special request basis using the **sar** command, or it can be saved automatically on a routine basis using the **sadc** command and the shell scripts **sa1** and **sa2.** The **sadc** command and the **sa1** and **sa2** shell scripts, collectively referred to as the System Activity Report Package, are used to sample, save, and process system activity data.

This chapter describes the commands and shell scripts in the System Activity Report Package.

sadc Command

The **sadc** command, the data collector, has the following format:

/usr/lib/sa/sadc *[t n] [ofile]*

The **sadc** command samples system data *n* times with an interval of *t* seconds *(t* should be greater than 5 seconds) between samples and writes in binary format to the file, *ofile,* or to standard output. If *t* and *n* are omitted, a special record is written.

The **sadc** command is used at system boot time, when booting to a multiuser state, to mark the time at which the counters restart from zero. For example, the **/etc/init.d/perf** file contains a command line that writes a special record to the daily data file, that marks the reboot of the system. This is done by the command entry:

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

sa1 Shell Script

The shell script **sa1,** a variant of the **sadc** command, has the following format:

/usr/lib/sa/sa1 *[t n]*

The **sa1** shell script is used to collect and store data in the binary file **/usr/adm/sa/sa***dd,* where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds. If these arguments are omitted, the records are written only one time.

To produce records every 20 minutes during working hours and every hour otherwise, add the following to the **/usr/spool/cron/crontabs/sys** file:

```
0 * * * 0,6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
```

To produce a record three times at 5 minute intervals in the binary file **/usr/adm/sa/sa***dd,* enter the following command:

```
$/usr/lib/sa/sa1 300 3<CR>
$
```

Note: To see the results of this command, on standard output, see the **sar** *command.*

sa2 Shell Script

The shell script **sa2,** a variant of the **sar** command, writes a daily report in the file **/usr/adm/sa/sar***dd.* The **sa2** shell script as the following format:

/usr/lib/sa/sa2 [-ubdycwaqvmprDSAC] [-s *time]* **[-e** *time*] **[-i** *sec]*

Note: See the **sar** command in Chapter 5, "Performance Tools" for an explanation of the options.

The **sa2** shell script writes a report to the daily report file **/usr/adm/sa/sar***dd* using options **[-ubdycwaqvmprDSAC].** The report starts at **-s** *time,* ends at **-e** *time,* and is taken as close to **-i** *seconds* as possible.

To report important activities during the working day on an hourly basis, add the following entry to the file **/usr/spool/cron/crontabs/sys:**

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

To write a report showing terminal device activity each hour from 1:00 p.m. to 4:00 p.m., enter the following command:

```
$/usr/lib/sa/sa2 -y -s13:00 -e16:O0 -i3600<CR>
$
```

Note: Nothing will be output on the screen, but the report will be added to the daily report file.

Chapter 5: Performance Tools

General

Internal system activity is measured by counters contained in the UNIX System kernel. Each time an operation is performed, an associated counter is incremented. The performance tools **(sar, sag, sadp,** and **timex** commands) allow you to monitor the values of these counters. Each of the commands that make up the performance tools are described in detail in this chapter. The following is a brief description of these commands:

sar	Calls **sadc** to sample cumulative activity counters in the operating system at specified intervals of time and saves the samples in binary format.
sag	Graphically displays the information collected by **sar.**
sadp	Produces profiles of disk access location and seek distance.
timex	Reports both system-wide and per-process activity during the execution of a command or program.

The functions monitored by the **sar** command are also discussed in this chapter.

The command responses found in this chapter are only examples. The values you receive may be different from the values used in these examples; it depends on your application or benchmark. When tuning your system, it is recommended that you use a benchmark or have the system under normal load for your application to allow you to tune directly toward your specific application.

sar Command

The **sar** command is the system activity reporter, it can be used either to gather system activity data or to extract what has been collected in data files created by **sa1** and **sa2** shell scripts. The **sa1** and **sa2** shell scripts are started by entries put in the **crontab sys** file.

The **sar** command has the following formats:

sar [-ubdycwaqvmprDSAC] [-o *file] t [n]*
sar [-ubdycwaqvmprDSAC] [-s *time]* **[-e** *time]* **[-i** *sec]* **[-f** *file]*

In the first format, **sar** samples cumulative activity counters in the operating system at intervals specified by *n* for a time (seconds) specified by *t (t* should be 5 seconds or greater). The default value of *n* is 1. If the **-o** option is specified, samples are saved in *file* in binary format.

In the second format, 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/sa***dd (dd* is the current day). The **-s** and **-e** options define the starting and ending times for the report. Starting and ending times are of the form *hh[:mm[:ss]]. the* **-i** option specifies, in seconds, the intervals to select records. If the **-i** option is not included, all intervals found in the data file are reported.

Most of the remainder of this chapter is devoted to describing the **sar** options plus an analysis of sample outputs of the options. The options **-D, -S,** and **-C,** which are Remote File Sharing options, are not described in this document. They are described under "Remote File Sharing" in the *AT&T 3B2 Computer, UNIX System V, System Administrator's Guide.*

Note: The **-C** option is only available on UNIX System Release 3.1 and later releases.

sar Command Options

sar -a Command

The **sar -a** command reports the use of file-access operations. The UNIX Operating System routines reported are as follows:

iget/s Number of files located by inode entry per second.

namei/s Number of file system path searches per second. The **namei** routine calls **iget,** so **iget/s** is always larger than **namei/s.**

dirbk/s Number of directory block reads issued per second.

The following is an example of **sar -a** output, with a 30-second sampling interval:

```
unix unix rel 2 3B2    4/16/88
12:41:40  igets/s name/s dirbk/s
12:42:10       4       1       3
12:42:40       2       1       1
12:43:10       5       2       3
Average        4       1       3
```

The **sar -a** output shows how heavily programs and applications are using the file system(s). The larger the values reported, the more time the UNIX System kernel is spending to access user files. The **-a** option is helpful for understanding the disk dependency of the application system. It is not used for any specific tuning step.

sar -b Command

The **sar -b** command reports the following buffer activity.

bread/s	Average number of physical blocks read into the system buffers from the disk (or other block devices) per second.
lread/s	Average number of logical blocks read from system buffers per second.
%rcache	Fraction of logical reads found in buffer cache (100 percent minus the ratio of **breads** to **lreads).**
bwrit/s	Average number of physical blocks written from the system buffers to disk (or other block devices) per second.
lwrit/s	Average number of logical blocks written to system buffers per second.
%wcache	Fraction of logical writes found in buffer cache (100 percent minus the ratio of **bwrit/s** to **lwrit/s).**
pread/s	Average number of physical read requests per second.
pwrit/s	Average number of physical write requests per second.

The entries that you should be most interested in are the cache hit ratios, **%rcache** and **%wcache,** which measure the effectiveness of system buffering. If **%rcache** falls below 90 or **%wcache** falls below 65, it may be possible to improve performance by increasing the number of buffers.

The following is an example of a **sar -b** output:

```
unix unix rel 2 3B2    4/16/88
16:32:57 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
16:33:07       3      39      93       1      16      91       0       0
16:33:17       4      40      90       2      16      87       0       0
16:33:27       4      41      90       3       7      64       0       0
Average        4      40      91       2      13      84       0       0
```

This example shows that the buffers are not causing any bottlenecks, because all data is within acceptable limits. On UNIX System Release 3.1 and later releases, a parallel option **(-Db)** is available for systems that have Remote File Sharing installed. See the *System Administrator's Guide* for more information on **sar** options used with Remote File Sharing.

sar -c Command

The **sar -c** command reports system calls in the following categories:

scall/s All types of system calls per second, generally about 30 per second on a busy 4- to 6-user system.

sread/s Read system calls per second.

swrit/s Write system calls per second.

fork/s Fork system calls per second, about 0.5 per second on a 4- to 6-user system. This number will increase if shell scripts are running.

exec/s Exec system calls per second. (If **[exec/s)** / **(fork/s)** is greater than 3, look for inefficient $PATHs.]

rchar/s Characters (bytes) transferred by read system calls per second.

wchar/s Characters (bytes) transferred bywrite system calls per second.

Typically, reads plus writes account for about half of the total system calls, although this varies greatly with the activities that are being performed by the system.

The following is an example of a **sar -c** output:

```
unix unix rel 2 3B2    4/16/88
18:33:04 scall/s sread/s  swrit/s  fork/s exec/s rchar/s wchar/s
18:33:35      38      16       6   0.03   0.03    6089    1638
18:34:05      39      16       4   0.07   0.07    6123    1602
18:34:35      38      17       5   0.17   0.17    6042    1704
Average       38      16       5   0.09   0.09    6085    1648
```

A parallel option **(-Dc)** is available for systems that have Remote File Sharing installed. See the *System Administrator's Guide* for more information on **sar** options used with Remote File Sharing.

sar -d Command

The **sar -d** command reports the activity of block devices.

device	Name of the block device(s) that **sar** is monitoring.
%busy	Percent of time the device was servicing a transfer request.
avque	The average number of requests outstanding during the period of time (measured only when the queue is occupied).
r+w/s	Number of read and write transfers to the device per second.
blks/s	Number of 512-byte blocks transferred to the device per second.
avwait	Average time in milliseconds that transfer requests wait idly in the queue (measured only when the queue is occupied).
avserv	Average time in milliseconds for a transfer request to be completed by the device (for disks this includes seek, rotational latency, and data transfer times).

The following two examples illustrate the output from **sar -d.** The first example is from a computer with a non-SCSI (Small Computer System Interface) integral disk, that is, a disk that does not use a SCSI interface. This example illustrates data being transferred from the hard disk (hdsk-0) to the floppy disk (fdsk-0).

The second example is from a computer with SCSI integral disks, that is, disks that use a SCSI interface. The example illustrates data being transferred from one SCSI hard disk (sd00-0) to another (sd00-1).

Note: The device name for a SCSI Release 2 hard disk is sd01.

Non-SCSI integral disk:

```
unix unix rel 2 3B2    4/16/88
13:46:28  device %busy  avque  r+w/s  blks/s  avwait  avserv
13:46:58  hdsk-0      6    1.6      3       5    13.8    23.7
          fdsk-0     93    2.1      2       4   467.8   444.0
13:47:28  hdsk-0     13    1.3      4       8    10.8    32.3
          fdsk-0    100    3.1      2       5   857.4   404.1
13:47:58  hdsk-0     17     .7      2      41      .6    48.1
          fdsk-0    100    4.4      2       6  1451.9   406.5
Average   hdsk-0     12    1.2      3      18     8.4    34.7
          fdsk-0     98    3.2      2       5   925.7   418.2
```

SCI integral disk:

```
unix unix rel 2 3B2    4/16/88
14:16:24  device %busy  avque  r+w/s  blks/s  avwait  avserv

14:16:52  sd00-0      2    1.0      1       3     0.0    17.9
          sd00-1      6    1.1      3       5     2.0    23.9

14:17:21  sd00-0      2    1.0      1       2     0.0    19.6
          sd00-1      6    1.1      3       5     0.2    24.3

14:17:48  sd00-0      3    1.0      1       3     0.3    18.3
          sd00-1      7    1.1      3       5     1.3    25.4

14:18:15  sd00-0      3    1.0      1       3     0.0    17.2
          sd00-1      5    1.0      2       5     0.0    21.6

Average   sd00-0      2    1.0      1       3     0.1    18.2
          sd00-1      6    1.0      3       5     0.9    23.8
```

Looking at these results tells us why hard disk technology has improved the performance of supermicrocomputers.

Note that queue lengths and wait times are measured while the queue has something on it. If the value of **%busy** is small, large queues and service times probably represent the periodic **sync** efforts by the system to ensure that altered blocks are written to the disk in a timely fashion.

sar -m Command

The **sar -m** command reports on inter-process communication activities. Message and semaphore calls are reported as follows:

msg/s Number of message operations (sends and receives) per second.

sema/s Number of semaphore operations per second.

The following is an example of a **sar -m** output:

```
unix unix rel 2 3B2    4/16/88
15:16:58   msg/s   sema/s
15:17:32    0.00    0.00
15:18:02    0.00    0.00
15:18:32    0.00    0.00
Average     0.00    0.00
```

These figures will usually be zero (0.00) unless you are running applications that use the message or semaphore features.

sar -q Command

The **sar -q** command reports the average queue length while the queue is occupied and the percent of time it is occupied.

runq-sz Run queue of processes in memory; typically, this should be less than 2. Consistently higher values mean you are CPU-bound.

%runocc The percentage of time the run queue is occupied; the larger this value, the better.

swpq-sz Swap queue of processes to be swapped out; the smaller this number, the better.

%swpocc The percentage of time the swap queue is occupied; the smaller this value, the better.

The following is an example of **sar -q** output:

```
unix unix rel 2 3B2    4/16/88
11:00:56 runq-sz %runocc swpq-sz %swpocc
11:01:07     1.7      98     1.5      36
11:01:17     1.0      63     1.0      31
11:01:27     1.0      58     1.0      49
Average      1.3      74     1.2      39
```

In this example, the processor utilization **(%runocc)** varies between 58 percent and 98 percent while the fraction of time the swap queue is not empty **(%swpocc)** is 31 percent to 49 percent. With these percentages, memory is not causing a major bottleneck in the system throughput, but more memory would help reduce the swapping/paging activity.

If **%runocc** is greater than 90 and **runq-sz** is greater than 2, the CPU is heavily loaded and response is degraded. In this case, additional CPU capacity may be required to obtain acceptable system response. If **%swpocc** is greater than 20, more memory or fewer buffers would help reduce swapping/paging activity.

sar -u Command

The **sar -u** command (default) lists the CPU utilization. At any given moment the processor will be either busy or idle. When busy, the processor will be in either user or system mode. When idle, the processor will either be waiting for input/output completion or has no work to do.

The **-u** option of **sar** lists the percent of time that the processor is in system mode **(%sys),** user mode **(%usr),** waiting for input/output completion **(%wio),** and idle time **(%idle).**

> **Note:** If your computer has one or more Multiprocessor Enhancement feature cards installed, the **-u** option displays the CPU utilization for both the main processor and the coprocessor(s). The displays, however, are not the same. A coprocessor display does not include waiting for input/output completion **(%wio),** but this display does report the number of system calls per second **(scail/s)** that occurred on a coprocessor.

In typical timesharing use, **%sys** and **%user** are about the same value. In special applications, either of these may be larger than the other without anything being abnormal. A high **%wio** generally means a disk bottleneck. On releases prior to Release 2.1, a high **%idle** with degraded response time may mean memory constraints; time spent waiting for memory is attributed to **%idle.**

The following is an example of executing **sar -u** on a computer without a Multiprocessor Enhancement feature card:

```
unix unix rel 2 3B2      04/12/88
10:02:07        %usr      %sys    %wio    %idle
10:02:27          82        18       0        0
10:02:47          39        35      16       10
10:03:07           7        28      15       50
10:03:27           1        16       0       83

Average           32        24       8       36
```

The next display is an example of executing **sar -u** on a computer with a Multiprocessor Enhancement feature card:

```
unix unix rel 2 3B2     04/12/88
10:02:07        %usr       %sys    %wio    %idle
10:02:27          82        18       0        0
10:02:47          39        35      16       10
10:03:07           7        28      15       50
10:03:27           1        16       0       83

Average           32        24       8       36


10:02:07        %co-usr   %co-sys         %co-idle        scall/s
10:02:27             98         0                2             0
10:02:47             65         0               35             0
10:03:07             11         0               89             0
10:03:27              0         0              100             0

Average              44         0               56             0
```

The next display is an example of executing **sar -u** on a computer with three Multiprocessor Enhancement feature cards:

```
unix unix rel 2 3B2     04/12/88
10:02:07        %usr      %sys    %wio    %idle
10:02:27          82        18       0        0
10:02:47          39        35      16       10
10:03:07           7        28      15       50
10:03:27           1        16       0       83

Average           32        24       8       36


10:02:07        %co-usr   %co-sys         %co-idle        scall/s
10:02:27             98         0                2              0
                     48         0               52              0
                     75         0               25              0
10:02:47             65         0               35              0
                    100         0                0              0
                      2         0               98              0
10:03:07              9         0               91              0
                     15         0               85              0
                     20         0               80              0
10:03:27              0         0              100              0
                      2         0               98              0
                     10         0               90              0

Average              37         0               63              0
```

A parallel option **(-Du)** is available for systems that have Remote File Sharing installed. See the *System Administrator's Guide* for a description of **sar** options that are used with Remote File Sharing.

sar -v Command

The **sar -v** command reports the status of process, inode, file, and shared memory record file tables. From this report you know when the system tables need to be modified.

proc-sz Number of process table entries now being used/allocated in the kernel.

inod-sz Number of inode table entries now being used/allocated in the kernel.

file-sz Number of file table entries now being used/allocated in the kernel.

ov Number of times an overflow occurred. (One column for each of the above three items.)

lock-sz The number of shared memory record table entries now being used/allocated in the kernel.

Note: Releases prior to 2.1 also include **text-sz** (number of text table entries being used/allocated in the kernel) and **fhdr-sz** (number of shared memory file table headers being used/allocated in the kernel).

The values are given as *number currently used/table size.* An example of the **sar -v** command follows:

```
unix unix rel 2 3B2     4/16/88
17:36:05  proc-sz ov   inod-sz ov   file-sz ov   lock-sz
17:36:35   17/ 40  0    39/ 80  0    29/ 80  0     0/ 50
17:37:05   19/ 40  0    46/ 80  0    35/ 80  0     0/ 50
17:37:35   18/ 40  0    43/ 80  0    34/ 80  0     0/ 50
```

This example shows that all tables are large enough to have no overflows. Sizes could be reduced to save main memory space if these are the highest values ever recorded.

sar -w Command

The **sar -w** command reports swapping and switching activity. The following are some target values and observations.

swpin/s Number of transfers into memory per second.

bswin/s Number of 512-byte block units (blocks) transferred for swap-ins (including initial loading of some programs) per second.

swpot/s Number of transfers from memory to the disk swap area per second. If greater than 1, memory may need to be increased or buffers decreased.

bswot/s Number of blocks transferred for swap-outs per second.

pswch/s Process switches per second. This should be 30 to 50 on a busy 4- to 6-user system.

The following is an example of a **sar -w** output:

```
unix unix rel 2 3B2     4/16/88
19:53:44 swpin/s bswin/s  swpot/s  bswot/s  pswch/s
19:53:58     0.0     0.0      0.0      0.0       37
19:54:14     0.0     0.0      0.0      0.0       39
19:54:24     0.0     0.0      0.0      0.0       39
Average      0.0     0.0      0.0      0.0       38
```

This example shows that there is enough memory for the currently active users, since no swapping is occurring.

sar -p Command

Note: The **-p** option does not apply to swapping releases.

The **sar -p** command reports paging activity. The following page rates are recorded.

vflt/s Number of address translation page faults per second (valid page not present in memory).

pflt/s Number of page faults from protection errors per second (illegal access to page) or "copy-on-writes." The **pflt/s** page rate generally consists entirely of "copy-on-writes."

pgfil/s Number of **vflt/s** per second satisfied by a page-in from the file system. (Each **pgfil** causes two **lreads;** see **sar -b.)**

rclm/s Number of valid pages per second that the system has reclaimed (added to list of free pages).

The following is an example of a **sar -p** output:

```
unix unix rel 2 3B2        4/16/88
12:01:51 vflt/s pflt/s pgfil/s rclm/s
12:56:52  13.91   2.80   5.63   11.21
```

sar -r Command

Note: The **-r** option does not apply to swapping releases.

The **sar -r** command records the number of memory pages and swap file disk blocks that are currently unused. The following are recorded:

freemem Instantaneous value of 2-kilobyte pages of memory available to user processes at sampling time.

freeswp Number of disk blocks available for process swapping.

An example of **sar -r** output follows:

```
unix unix rel 2 3B2        4/16/88
12:01:51 freemem  freeswp
12:56:52     208     5848
```

sar -y Command

The **sar -y** command reports terminal device activities. If you have much terminal input/output, you can use this report to determine if there are any bad lines. Activities recorded are defined as follows:

rawch/s Input characters (raw queue) per second.

canch/s Input characters processed by canon (canonical queue) per second.

outch/s Output characters (output queue) per second.

rcvin/s Receiver hardware interrupts per second.

xmtin/s Transmitter hardware interrupts per second.

mdmin/s Modem interrupts per second.

The number of modem interrupts per second **(mdmin/s)** should be close to 0, and the receive and transmit interrupts per second **(xmtin/s** and **rcvin/s)** should be less than or equal to the number of incoming or outgoing characters, respectively. If this is not the case, check for bad lines.

The following is an example of **sar -y** output:

```
unix unix rel 2 3B2       4/16/88
16:50:11 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
16:50:41     112      15     653     103     102       0
16:51:11     107       7     654     104     105       0
16:51:41      99       5     641      99     105       0
Average      106       9     649     102     104       0
```

sar -A Command

The **sar -A** command provides an overall view of system performance. Use it to get a more global perspective. All the options that are available with your version of the System Performance Analysis Utilities are reported. If Remote File Sharing is installed, the report also includes the Remote File Sharing options **-C, -D,** and **-S.** These options are described in the *System Administrator's Guide.*

If data from more that one time slice is shown, the report includes averages. The following is an example of **sar -A.**

```
unix unix rel 2 3B2     4/16/88

11:26:06    %usr    %sys    %sys    %wio   %idle
                   local  remote
11:26:08       2       3       0       0      95
11:26:06  device   %busy   avque   r+w/s  blks/s  avwait  avserv
11:26:08 hdsk-0          0     1.5       0       0    10.9    21.1
         hdsk-1          0     2.1       0       0    30.6    27.9
11:26:06 runq-sz %runocc swpq-sz %swpocc
11:26:08     1.1      83
11:26:00 bread/s  lread/s %rcache  bwrit/s lwrit/s %wcache pread/s pwrit/s
11:26:06
   local     0         0     100       0       1     100       0       0
  remote     0         0       0       0       0       0
11:26:06 swpin/s  bswin/s  swpot/s  bswot/s pswch/s
11:26:08    0.00     0.0    0.00     0.0       2
11:26:06 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
11:26:08
   in          0       0       0            0.00       0       0
   out         0       0       0            0.00       0       0
   local      30      12       1    0.00    0.00   22860     483
11:26:06  iget/s namei/s dirbk/s
11:26:08       0       0       0
11:26:06 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
11:26:08       0       0       0       0       0       0
11:26:06 proc-sz ov inod-sz ov file-sz ov lock-sz
11:26:08  17/ 60  0  56/150  0  26/150  0   0/100
11:26:06   msg/s  sema/s
11:26:08    0.00    0.00
11:26:06  vflt/s  pflt/s pgfil/s  rclm/s
11:26:08    4.13    0.83    0.00    0.00
11:26:06 freemem freeswp
11:26:08     899   10104
11:26:06  snd-inv/s  snd-msg/s   rcv-inv/s  rcv-msg/s dis-bread/s blk-inv/s
11:26:08       0.0        0.0         0.0        0.0        0.0 0.0
11:26:06 serv/lo-hi   request   request    server    server
              3 - 6     %busy  avg lgth    %avail avg avail
11:26:08          0       0.0         0       0.0         0
```

sag Command

The **sag** command graphically displays the system activity data stored in a binary data file by a previous **sar** run. Any of the **sar** data items may be plotted separately or in combination. The **sag** command invokes **sar** and matches strings in the data column header.

The **sag** command has the following format:

sag *[options]*

The options for **sag** are listed below:

-s *time*	Select data later than **-s** *time* in the form of *hh[:mm].* The default of *time* is 08:00.
-e *time*	Select data up to an **-e** *time.* Default is 18:00.
-i *sec*	Select data at intervals as close as possible to **-i** *sec* seconds.
-f *file*	Use **-f** *file* as the data source for **sar.** Default is the current daily data file **(/usr/adm/sa/sadd).**
-T *term*	Produce output suitable for terminal *term.* See *tplot* for known terminals. The default for *term* is $TERM.
-x *spec*	The x axis specification *spec* is in the form: **name** *[op name]...[lo hi].* (Name is either a string that will match a column header in the **sar** report or a number value.)
-y *spec*	The **y** axis specification *spec* is in the form: **name** *[op name]...[lo hi].*

Figure 5-1 shows a typical **sag** display. Running **sar** will show what data are available.

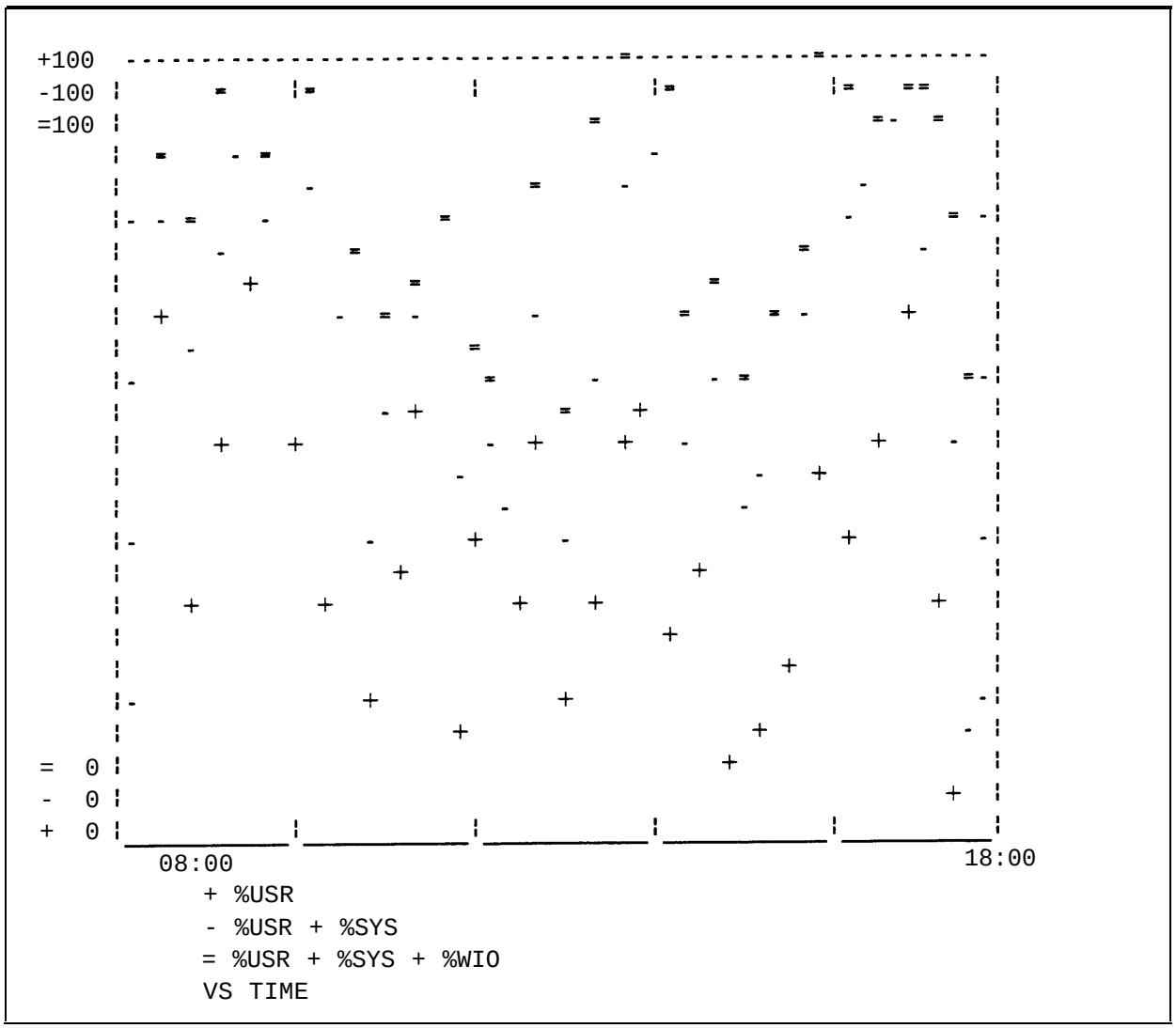


Figure 5-1: Example of **sag** Output

In Figure 5-1, the processor is completely utilized over three time intervals:

9 a.m. to 10 a.m.
1 p.m. to 2 p.m.
3:30 p.m. to 5:30 p.m.

Remember the fraction of time that the processor is busy is the sum of user **(%usr)** mode time and system **(%sys)** mode time. When this sum approaches 100 percent, the processor is running at its maximum capacity as configured. The sum of **%usr** + **%sys** + **%wio** is about the same as the sum of **%usr** + **%sys (%wio** is low). This means that the disk subsystem is able to handle all requests that the processor generates with little delay. From this example, the first place to look to reduce any bottleneck is in reducing processor load.

The **sag** command is useful only if you have a standard output device that can read plotting instructions. Refer to **tplot**(1G) in the *AT&T UNIX System V, User's Reference Manual* for a list of terminals with this capability.

timex Command

The **timex** command times a command and reports the system activities that occurred during the time the command was executing. If no other programs are running, then **timex** can give you a good idea of which resources a specific command uses during its execution. System consumption can be collected for each application program and used for tuning the heavily loaded resources.

The **timex** command has the following format:

timex *[-p -o -s] command*

The **-p** and **-o** options only work if System Accounting is operational on the system. The **-s** option reports the total system activity, not just what was called for in *command* but everything that occurred during the execution interval of *command.* All the data items listed in **sar** are reported.

In the following example, the **date** command is used. While **date** was used in the example for its simplicity, it is not the best example since it is not a major user of system resources.

The **timex** command can also be used in the following way:

$ timex -s *application program*

Your application program will operate normally. When you finish and exit, the **timex** result will be printed on your screen. This can be extremely interesting since you get a clear picture of system resources used by your program.

```
$ timex -s date
Thur Apr 16 08:32:50 EDT 1988
real        0.18
user        0.01
sys         0.12
unix unix rel 2 3B2    04/16/88
08:32:05    %usr     %sys    %sys    %wio   %idle
                     local  remote
08:32:05       5      79       0      16       0
08:32:05 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
   local     7      122       94       0       16      100       0       0
   remote    0        0        0       0        0        0
08:32:05  device   %busy   avque   r+w/s  blks/s  avwait  avserv
08:32:05 hdsk-0          9     1.0       3       7     0.0    25.0
         hdsk-1         12     1.0       3       7     0.0    35.0
08:32:05 rawch/s  canch/s  outch/s  rcvin/s xmtin/s mdmin/s
08:32:05       0        0       52       0       4       0
08:32:05 scall/s sread/s  swrit/s  fork/s  exec/s rchar/s wchar/s
08:32:05
   in          0       0       0            0.00       0       0
   out         0       0       0            0.00       0       0
   local     231      31       3    5.17    6.90   48703    1057
08:32:05 swpin/s  bswin/s  swpot/s bswot/s pswch/s
08:32:05    0.00     0.0    0.00     0.0      22
08:32:05  iget/s namei/s  dirbk/s
08:32:05      36      19      38
08:32:05 runq-sz %runocc  swpq-sz  %swpocc
08:32:05
08:32:05 proc-sz ov inod-sz ov file-sz ov  lock-sz
08:32:05  19/ 60  0  56/150  0  26/150  0   0/100
08:32:05   msg/s  sema/s
08:32:05    0.00    0.00
08:32:05  vflt/s  pflt/s pgfil/s  rclm/s
08:32:05   50.00   39.66    3.45    0.00
08:32:05 freemem freeswp
08:32:05     871   10104
11:26:06  snd-inv/s  snd-msg/s   rcv-inv/s  rcv-msg/s dis-bread/s blk-inv/s
11:26:08       0.0        0.0         0.0        0.0        0.0   0.0
08:32:05 serv/lo-hi   request   request    server    server
               3 - 6     %busy  avg lgth    %avail avg avail
08:32:05          0       0.0         0       0.0         0
```

sadp Command

Note: The **sadp** command does not work with SCSI Release 1 Disk Modules.

The **sadp** command reports disk access locations and seek distance in tabular or histogram form. Disk activity is sampled once every second during the specified sampling interval. The length of the sampling interval and the number of reports printed during that interval are specified in the command line. Cylinder usage and seek distance are recorded in units of 20 cylinders.

The **sadp** command, the disk access profiler, has the following format:

sadp [-th] [-d *device[-drive]] s [n]*

The **-t** flag causes the data to be reported in tabular form. The **-h** flag causes the data to be reported in histogram form on a printer. If **-t** or **-h** is not specified, the report will be in tabular form. The **-d** option may be omitted, if only one device is present. The **sadp** command can only profile one device at a time.

Valid names for *device* are **hdsk** for non-SCSI integral disks, **sdsk** for SCSI integral disks, and **fdsk** for an integral floppy disk.

Drive specifies the disk drives and it may be a single drive number, two numbers separated by a dash to show an inclusive range, or a list of drive numbers separated by commas.

The **s** argument specifies the duration of the sampling interval in seconds. The sampling interval must be 10 seconds or greater. The **n** argument specifies the number of reports to be generated during the sampling interval. The default of **n** is 1.

An example of **sadp** output for SCSI hard disk drive 0 is illustrated in Figures 5-2 and 5-3.

Using the **sadp** output along with the output of **/etc/mount** (1M) or **/etc/prtvtoc** (lM) and a table of disk sections [see the *System Administrator's Guide* for the default partitioning of your disk(s)], you can identify the file systems with a large amount of I/O activity. In general, try to move areas of high activity close together. This will reduce the number of seeks over large distances.

The first graph (Figure 5-2) shows excellent disk cylinder locality of references. This means that every time the location of a block was referenced by the disk head (for reading or writing), it mostly occurred in the same general region of the disk. In the example, most references (as indicated by percent times referenced) occur for files near cylinders 450 to 600, with a few for files around cylinder 250. There are a few references to other files on the disk, but they occur only a small percent of the time. This graph shows, then, that the most often used files are grouped together in the same general region of cylinders on the disk; the more clustered the stars on the histogram, the better. Another way to say this is that the disk has an excellent file system configuration.

The second graph (Figure 5-3) shows another aspect of an excellent file system configuration: the head seek distance. Head seek distance is the distance the disk head has to move from the current cylinder to the cylinder of the next block referenced. In the example, most physical seeks are under ten cylinders. Specifically, some 14 percent of the seeks occurred within a distance of 0 to 8 cylinders, and some 17 percent of the seeks occurred within a distance of 8 to 16 cylinders. This means that for about one-third of the disk activity, the disk head was forced to move no more that 16 cylinders to reference a given block, and the more to the left the stars are grouped, the better.

These two graphs give an idea of how finely you can tune your system. If, after a working period of weeks or months, you can identify the file systems that are consistently the most active, you might consider repartitioning your disks to achieve the maximum from disk access activity. (See the *System Administrator's Guide* for more information.)

$ sadp -h -d sdsk-0 3600<CR>

Mon Apr 4 07:18:54 1988

unix unix 3.2.1 3 3B2

CYLINDER ACCESS HISTOGRAM

sdsk-0:
Total transfers = 33291

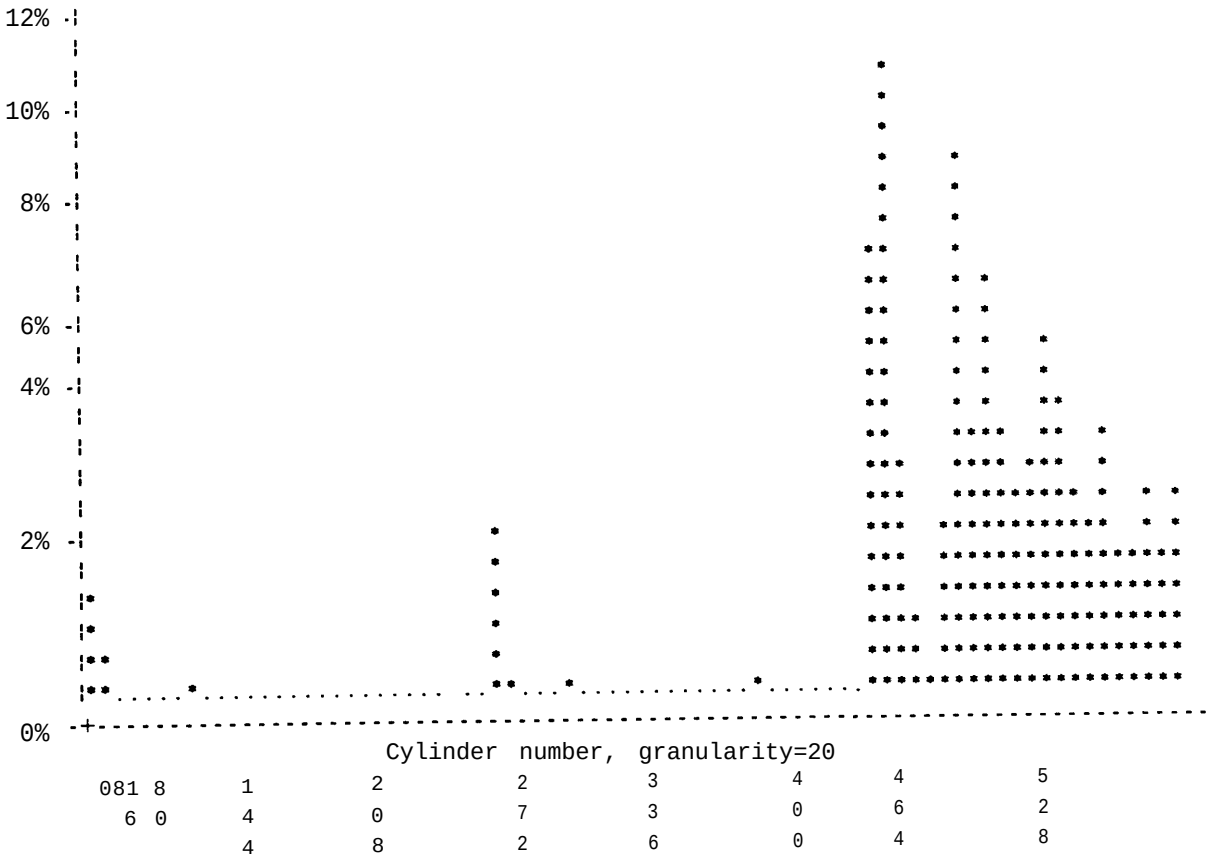


Figure 5-2: Output From **sapd:** Cylinder Access Histogram

```
SEEK DISTANCE HISTOGRAM

sdsk-0:
Total seeks = 30308
     ¦
20% -¦
     ¦
     ¦ *
16% -¦ *
     ¦ *
     ¦**
12% -¦**
     ¦**
 8% -¦**
     ¦**
     ¦***
     ¦***   *
     ¦**** **
 4% -¦*********
     ¦*********
     ¦************
     ¦**************
     ¦******************
     ¦**********************
     ¦*************************...*.............................*.............
 0% -+-------------------------------------------------------------------------
      =<< <        <     <     <     <     <     <     <
      081 8        1     2     2     3     4     4     5
        6 0        4     0     7     3     0     6     2
                   4     8     2     6     0     4     8
```

Figure 5-3: Output From **sadp:** Seek Distance Histogram

Index

P

R

S