THE EXPERT'S VOICE® IN ORACLE

Linux Recipes for Oracle DBAs

Real-world solutions for the intersection of Linux and Oracle technologies.

Darl Kuhn, Charles Kim, and Bernard Lopuz

Apress®

Linux Recipes for Oracle DBAs

ISBN-13 (pbk): 978-1-4302-1575-2

ISBN-13 (electronic): 978-1-4302-1576-9

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jonathan Gennick
Technical Reviewers: Bernard Lopuz, Charles Kim
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Kylie Johnston
Copy Editor: Kim Wimpsett
Associate Production Director: Kari Brooks-Copony
Production Editor: Elizabeth Berry
Compositor: Susan Glinert Stevens
Proofreader: Nancy Sixsmith
Indexer: Carol Burbo
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The source code for this book is available to readers at http://www.apress.com.

Contents

About the Authors

DARL KUHN is currently a DBA with Sun Microsystems. He has coauthored two other books: *RMAN Recipes for Oracle Database 11g: A Problem-Solution Approach (Apress, 2007)* and *Oracle RMAN Pocket Reference (O'Reilly, 2001)*. He also teaches advanced database courses at Regis University and performs volunteer database administration work for the Rocky Mountain Oracle Users Group. He has a graduate degree from Colorado State University and currently lives near Spanish Peaks, Colorado, with his wife, Heidi, and daughters, Brandi and Lisa.

CHARLES KIM serves as the practice manager of database technologies at Novara Solutions. He has more than 18 years of IT experience and has worked with Oracle since 1991. Charles is an Oracle ACE, coauthor of *Oracle Database 11*g *New Features for DBAs and Developers* (Apress, 2007), and author of the "Maximum Availability Architecture" case study at Oracle's web site (`http://www.oracle.com/technology/deploy/availability/htdocs/FNF_CaseStudy.html`); he has certifications in Oracle, Red Hat Linux, and Microsoft. Prior to Novara Solutions, Charles functioned as the chief Oracle database engineering counsel for Fidelity National Information Services and worked at companies such as GMAC Mortgage, Oracle Corporation, and i2 Technologies.

Charles has presented advanced topics for IOUG and Oracle OpenWorld on such topics as RAC/ASM and 24/7 high availability considerations. Charles also blogs regularly at `http://blog.dbaexpert.com` and provides technical solutions to Oracle DBAs and developers.

BERNARD LOPUZ is currently a senior technical support analyst at Oracle Corporation. In the early years of his IT career before he became an Oracle database administrator, he was a programmer developing Unisys Linc and Oracle applications, as well as interactive voice response (IVR) applications such as telephone banking voice-processing applications. He has wide experience using Red Hat AS and Oracle Enterprise Linux (OEL). Bernard was the technical reviewer of *RMAN Recipes for Oracle Database 11g: A Problem-Solution Approach* (Apress, 2007) and is an Oracle Certified Professional (OCP). He is pursuing a master's degree in computer information technology at Regis University in Denver, Colorado, and has a bachelor's degree in computer engineering from the Mapúa Institute of Technology in Manila, Philippines. Bernard lives in Ottawa, Canada, with his wife, Leizle, and daughters, Juliet and Carol.

CHAPTER 8

Analyzing Server Performance

The delineation of tasks between a system administrator and a DBA is often nebulous. This can be especially true in small shops where you find yourself wearing multiple hats. Even in large organizations with established roles and responsibilities, you'll still find yourself in an occasional "all-hands-on-deck" fire drill where you're expected to troubleshoot server issues. In these scenarios, you must be familiar with the operating system commands used to extract information from the server. An expert DBA does not diagnose database problems in a vacuum; you have to be server savvy.

Whenever there are application performance issues or availability problems, seemingly (from the DBA's perspective) the first question asked is, what's wrong with the database? Regardless of the source of the problem, the onus is often on the DBA to either prove or disprove whether the database is behaving well. This process sometimes includes determining server bottlenecks. The database and server have a symbiotic relationship. DBAs need to be well versed with techniques to monitor server activity.

This chapter covers techniques used to analyze the server's CPU, memory, disk I/O, and network performance. Take some time to familiarize yourself with the relevant commands covered in each section. Being able to quickly survey system activity will vastly broaden your database administrator skill set.

System administrators also heavily use the tools described in this chapter. Table 8-1 summarizes the operating system utilities described in this chapter. Being familiar with how these operating system commands work and how to interpret the output will allow you to better work in tandem with your system administration team when diagnosing server performance issues.

Table 8-1. *Performance and Monitoring Utilities*

Tool	Purpose
`vmstat`	Monitors processes, CPU, memory, or disk I/O bottlenecks.
`watch`	Periodically runs another command.
`ps`	Identifies highest CPU- and memory-consuming sessions. Used to identify Oracle sessions consuming the most system resources.
`top`	Identifies sessions consuming the most resources.
`mpstat`	Reports CPU statistics.
`sar`	Displays CPU, memory, disk I/O, and network usage, both current and historical.

Table 8-1. *Performance and Monitoring Utilities (Continued)*

Tool	Purpose
`free`	Displays free and used memory.
`df`	Reports on free disk space.
`du`	Displays disk usage.
`iostat`	Displays disk I/O statistics.
`netstat`	Reports on network statistics.

■ **Note** Oracle recommends you install the `sysstat` package on your database server. This package includes performance-monitoring utilities such as `mpstat`, `iostat`, and `sar`. Several of the recipes in this chapter utilize these tools. See Chapter 10 for details on installing the `sysstat` package.

8-1. Identifying System Bottlenecks

Problem

The application users are reporting that the database seems slow. You want to determine whether there are any system resource bottlenecks on the database server.

Solution

The `vmstat` (virtual memory statistics) tool is intended to help you quickly identify bottlenecks on your server. The `vmstat` command displays real-time performance information about processes, memory, paging, disk I/O, and CPU usage. This example shows using `vmstat` to display the default output with no options specified:

```
$ vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache    si   so    bi    bo   in    cs us sy id wa
14  0  52340  25272   3068 1662704    0    0    63    76    9    31 15  1 84  0
```

Here are some general heuristics you can use when interpreting the output of `vmstat`:

- If the `wa` (time waiting for I/O) column is high, this is usually an indication that the storage subsystem is overloaded. See recipes 8-9 and 8-10 for identifying the sources of I/O contention.
- If `b` (processes sleeping) is consistently greater than 0, then you may not have enough CPU processing power. See recipe 8-2 for identifying Oracle processes and SQL statements consuming the most CPU.
- If `so` (memory swapped out to disk) and `si` (memory swapped in from disk) are consistently greater than 0, you may have a memory bottleneck. See recipe 8-5 for details on identifying Oracle processes and SQL statements consuming the most memory.

■ **Note** The Linux `vmstat` command does not count itself as a currently running process.

How It Works

If your database server seems sluggish, then analyze the `vmstat` output to determine where the resources are being consumed. Table 8-2 details the meanings of the columns displayed in the default output of `vmstat`.

Table 8-2. *Column Descriptions of `vmstat` Output*

Column	Description
`r`	Number of processes waiting for runtime
`b`	Number of processes in uninterruptible sleep
`swpd`	Total virtual memory (swap) in use (KB)
`free`	Total idle memory (KB)
`buff`	Total memory used as buffers (KB)
`cache`	Total memory used as cache (KB)
`si`	Memory swapped in from disk (KB/s)
`so`	Memory swapped out to disk (KB/s)
`bi`	Blocks read in (blocks/s) from block device
`bo`	Blocks written out (blocks/s) per second to block device
`in`	Interrupts per second
`cs`	Context switches per second
`us`	User-level code time as a percentage of total CPU time
`sy`	System-level code time as a percentage of total CPU time
`id`	Idle time as a percentage of total CPU time
`wa`	Time waiting for I/O completion

By default, only one line of server statistics is displayed when running `vmstat` (without supplying any options). This one line of output displays average statistics calculated from the last time the system was rebooted. This is fine for a quick snapshot. However, if you want to gather metrics over a period of time, then use `vmstat` with this syntax:

```
$ vmstat <interval in seconds> <number of intervals>
```

While in this mode, `vmstat` reports statistics sampling from one interval to the next. For example, if you wanted to report system statistics every two seconds for ten intervals, then issue this command:

```
$ vmstat 2 10
```

You can also send the `vmstat` output to a file. This is useful for analyzing historical performance over a period of time. This example samples statistics every 5 seconds for a total of 60 reports and records the output in a file:

```
$ vmstat 5 60 > vmout.perf
```

Another useful way to use `vmstat` is with the `watch` tool. The `watch` command is used to execute another program on a periodic basis. This example uses `watch` to run the `vmstat` command every five seconds and to highlight on the screen any differences between each snapshot:

```
$ watch -n 5 -d vmstat
Every 5.0s: vmstat                                    Thu Aug  9 13:27:57 2007
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache  si   so    bi    bo   in    cs us sy id wa
 0  0    144  15900  64620 1655100   0    0     1     7   16     4  0  0 99  0
```

When running `vmstat` in `watch -d` (differences) mode, you'll visually see changes on your screen as they alter from snapshot to snapshot. To exit from `watch`, press Ctrl+C.

One last note, the default unit of measure for the memory columns of `vmstat` is in kilobytes. If you want to view memory statistics in megabytes, then use the `-S m` (statistics in megabytes) option:

```
$ vmstat -S m
```

OS WATCHER

Oracle provides a collection of Linux/Unix scripts that gather and store metrics for CPU, memory, disk, and network usage. The OS Watcher tool suite automates the gathering of statistics using tools such as `top`, `vmstat`, `iostat`, `mpstat`, `netstat`, and `traceroute`. If you don't have these utilities installed, see Chapter 10 for details on installing the `sysstat` package.

You can obtain OS Watcher from Oracle's MetaLink web site. Search for document ID 301137.1 or for the document titled "OS Watcher User Guide." Navigate to the Contents page, and search for the Download link.

This utility also has an optional graphical component for visually displaying performance metrics. The OS Watcher utility is currently supported on the following platforms: Linux, Solaris, AIX, Tru64, and HP-UX.

8-2. Identifying CPU-Intensive Processes

Problem

You want to identify which Oracle session is consuming the most CPU on the database server. If it's an Oracle session running a SQL query, then you want to display the associated SQL.

Solution

Use the `ps` command to identify the process IDs of sessions consuming the most CPU on the server. This next `ps` command displays the top ten CPU-consuming statements and the associated process IDs:

```
$ ps -e -o pcpu,pid,user,tty,args | sort -n -k 1 -r | head
```

To limit the output to `oracle` processes, use this command:

```
$ ps -e -o pcpu,pid,user,tty,args | grep -i oracle | sort -n -k 1 -r | head
```

Here is a partial listing of the output:

```
99.6 15940 oracle ? oracleRMDB1 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
74.5 16022 oracle ? oracleRMDB1 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
 3.8 16014 oracle ? rman
 1.2 16019 oracle ? oracleRMDB1 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
 0.1 16026 oracle  pts/2    -bash
 0.1 16021 oracle ? oracleRMDB1 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
```

The first column is the percentage of CPU being consumed. The second column is the process ID. You can use the process ID from the previous output as an input to the following query to show information about the Oracle session:

```
SET LINESIZE 80 HEADING OFF FEEDBACK OFF
SELECT
  RPAD('USERNAME : ' || s.username, 80) ||
  RPAD('OSUSER   : ' || s.osuser, 80) ||
  RPAD('PROGRAM  : ' || s.program, 80) ||
  RPAD('SPID     : ' || p.spid, 80) ||
  RPAD('SID      : ' || s.sid, 80) ||
  RPAD('SERIAL#  : ' || s.serial#, 80) ||
  RPAD('MACHINE  : ' || s.machine, 80) ||
  RPAD('TERMINAL : ' || s.terminal, 80)
FROM v$session s,
     v$process p
WHERE s.paddr = p.addr
AND   p.spid  = '&PID_FROM_OS';
```

If you run the prior query and supply to it the process ID of 15940, you get the following output:

```
USERNAME : INVMGR
OSUSER   : oracle
PROGRAM  : sqlplus@rmugprd.rmug.com (TNS V1-V3)
SPID     : 15940
SID      : 529
SERIAL#  : 2564
MACHINE  : rmugprd.rmug.com
TERMINAL :
```

From the prior output, it's a SQL*Plus session that is consuming the most CPU resources. To identify the SQL statement that this process is running, you pass to this query the operating system process ID as input:

```
SET LINESIZE 80 HEADING OFF FEEDBACK OFF
SELECT
  RPAD('USERNAME : ' || s.username, 80) ||
  RPAD('OSUSER   : ' || s.osuser, 80) ||
  RPAD('PROGRAM  : ' || s.program, 80) ||
  RPAD('SPID     : ' || p.spid, 80) ||
  RPAD('SID      : ' || s.sid, 80) ||
  RPAD('SERIAL#  : ' || s.serial#, 80) ||
  RPAD('MACHINE  : ' || s.machine, 80) ||
  RPAD('TERMINAL : ' || s.terminal, 80) ||
  RPAD('SQL TEXT : ' || q.sql_text, 80)
FROM v$session s
    ,v$process p
    ,v$sql     q
WHERE s.paddr          = p.addr
AND   p.spid           = '&PID_FROM_OS'
AND   s.sql_address    = q.address
AND   s.sql_hash_value = q.hash_value;
```

If you run the previous query for the process ID of 15940, you get the following output:

```
USERNAME : INVMGR
OSUSER   : oracle
PROGRAM  : sqlplus@rmugprd.rmug.com (TNS V1-V3)
SPID     : 15940
SID      : 529
SERIAL#  : 2564
MACHINE  : rmugprd.rmug.com
TERMINAL :
SQL TEXT : select  count(*) ,object_name from dba_objects,dba_segments
```

The previous queries in this solution allow you to quickly identify Oracle processes and SQL statements that are currently consuming the greatest CPU resources on your database server.

How It Works

When you run multiple databases on one server and are experiencing server performance issues, it can be difficult to identify which database and session are consuming the most system resources. In these situations, use the `ps` command to identify the highest-consuming process and correlate that to a database session.

Once you have identified the highest resource-consuming session, then you have the option of trying to tune the operation (whether it be SQL, RMAN, and so on), or you might want to terminate the process. See recipe 3-2 for details on how to kill a Linux process and/or stop a SQL session.

Another tool for identifying resource-intensive processes is the `top` command. Use this utility to quickly identify which processes are the highest consumers of resources on the server. By default, `top` will repetitively refresh (every three seconds) information regarding the most CPU-intensive processes. Here's the simplest way to run `top`:

```
$ top
```

Here's a fragment of the output:

```
top - 08:58:33 up 4 days, 13:30,  2 users,  load average: 20.52, 20.58, 19.79
Tasks: 129 total,  22 running, 107 sleeping,   0 stopped,   0 zombie
Cpu(s): 95.2% us,  4.8% sy,  0.0% ni,  0.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   2074904k total,  2045824k used,    29080k free,     3236k buffers
Swap:  4184924k total,    52512k used,  4132412k free,  1580060k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1446 oracle    25   0  499m  26m  19m R   10  1.3   2:58.61 oracle
 1465 oracle    25   0  499m  26m  19m R   10  1.3   2:55.71 oracle
 1708 oracle    25   0  497m  23m  19m R   10  1.2   2:48.57 oracle
23444 oracle    25   0  539m  56m  21m R   10  2.8  20:33.85 oracle
23479 oracle    25   0  539m  56m  21m R   10  2.8  20:24.11 oracle
23499 oracle    25   0  515m  40m  21m R   10  2.0  20:34.25 oracle
```

The process IDs of the top-consuming sessions are listed in the first column (PID). Use the SQL queries in the "Solution" section of this recipe to map the operating system process ID to information in the Oracle data dictionary.

While `top` is running, you can interactively change its output. For example, if you type `>`, this will move the column that `top` is sorting one position to the right. Table 8-3 lists some key features that you can use to alter the `top` display to the desired format.

Table 8-3. *Commands to Interactively Change the `top` Output*

Command	Function
Spacebar	Immediately refreshes the output.
< or >	Moves the sort column one to the left or to the right. By default, `top` sorts on the CPU column.
`d`	Changes the refresh time.
`R`	Reverses the sort order.
`z`	Toggles the color output.
`h`	Displays help menu.
`F` or `O`	Chooses a sort column.

Type `q` or press Ctrl+C to exit `top`. Table 8-4 describes several of the columns displayed in the default output of `top`.

Table 8-4. *Column Descriptions of the `top` Output*

Column	Description
`PID`	Unique process identifier.
`USER`	OS username running the process.
`PR`	Priority of the process.
`NI`	Nice value or process. Negative value means high priority. Positive value means low priority.
`VIRT`	Total virtual memory used by process.
`RES`	Nonswapped physical memory used.
`SHR`	Shared memory used by process.
`S`	Process status.
`%CPU`	Processes percent of CPU consumption since last screen refresh.
`%MEM`	Percent of physical memory the process is consuming.
`TIME`	Total CPU time used by process.
`TIME+`	Total CPU time, showing hundredths of seconds.
`COMMAND`	Command line used to start a process.

You can also run `top` using the `-b` (batch mode) option and send the output to a file for later analysis:

```
$ top -b > tophat.out
```

While running in batch mode, the `top` command will run until you kill it (with a Ctrl+C) or until it reaches a specified number of iterations. You could run the previous `top` command in batch mode with a combination of `nohup` and `&` to keep it running regardless if you were logged onto the system. The danger there is that you might forget about it and eventually create a very large output file (and an angry system administrator).

If you have a particular process that you're interesting in monitoring, use the `-p` option to monitor a process ID or the `-U` option to monitor a specific username. You can also specify a delay and number of iterations by using the `-d` and `-n` options. The following example monitors the `oracle` user with a delay of 5 seconds for 25 iterations:

```
$ top -U oracle -d 5 -n 25
```

■ **Tip** Use the `man top` or `top --help` commands to list all the options available with your operating system version.

USING THE /PROC/<PID> FILES TO MONITOR PROCESS ACTIVITY

For every Linux process that is running, a directory is created in the `/proc` virtual filesystem. For example, say you want to view some details about the operating process ID of 19576. You can navigate to the virtual `/proc/19576` directory and do a long listing. You see several informational files and directories related to this running process:

```
$ cd /proc/19576
$ ls -l
```

Here is a partial listing of the output:

```
-r--r--r--  1 oracle oinstall 0 Jul  4 13:30 cmdline
lrwxrwxrwx  1 oracle oinstall 0 Jul  4 13:31 cwd -> /oracle/product/10.2/dbs
-r--------  1 oracle oinstall 0 Jul  4 13:31 environ
lrwxrwxrwx  1 oracle oinstall 0 Jul  4 13:31 exe
/oracle/product/10.2/bin/oracle
dr-x------  2 oracle oinstall 0 Jul  4 13:32 fd
-rw-r--r--  1 oracle oinstall 0 Jul  4 13:31 loginuid
-r--r--r--  1 oracle oinstall 0 Jul  4 13:31 maps
-r--r--r--  1 oracle oinstall 0 Jul  4 13:31 status
-rw-------  1 oracle oinstall 0 Jul  4 13:31 mem
```

The output tells us that this is an `oracle` process, and now you can analyze it further by looking at the memory usage `maps` file or the `status` file. Since these files do not exist on disk, use a utility such as `cat` to display their contents:

```
$ cat /proc/<PID>/maps
$ cat /proc/<PID>/status
```

8-3. Identifying CPU Bottlenecks

Problem

You want to monitor the system load on your CPUs.

Solution

As a DBA, you'll also need to periodically examine the load on CPUs to determine system bottlenecks. The `mpstat` (multiple processor statistics) utility displays statistics for processors on the server:

```
$ mpstat
Linux 2.6.9-55.0.6.ELsmp (rmugprd.rmug.com)  07/04/2008
12:39:52 PM  CPU   %user   %nice %system %iowait    %irq   %soft   %idle    intr/s
12:39:52 PM  all   35.21    0.06    0.71    0.24    0.00    0.00   63.78   1008.87
```

The default output of `mpstat` will show only one line of aggregated statistics for all CPUs on the server. You can also view CPU snapshots that report statistics accumulated between intervals. The following example uses the `-P` option to report only on processor 0; it displays output every 2 seconds for a total of 20 different reports:

```
$ mpstat -P 0 2 20
```

Here are a few lines of the output:

```
12:38:14 PM  CPU   %user   %nice %system %iowait    %irq   %soft   %idle    intr/s
12:38:16 PM    0   92.00    0.00    8.00    0.00    0.00    0.00    0.00   1002.50
12:38:18 PM    0   97.50    0.00    2.50    0.00    0.00    0.00    0.00   1002.00
12:38:20 PM    0   96.00    0.00    4.00    0.00    0.00    0.00    0.00   1002.50
12:38:22 PM    0   93.53    0.00    6.47    0.00    0.00    0.00    0.00    872.14
```

See Table 8-5 under "How It Works" for descriptions of the `mpstat` output. Here are some general guidelines for interpreting the output of the previous report:

- If `%idle` is high, then your CPUs are most likely not overburdened.
- If the `%iowait` output is a nonzero number, then you may have some disk I/O contention.
- If you identify that the CPUs are overloaded, see recipe 8-2 for techniques to pinpoint sessions consuming the most processor resources.

How It Works

Use the `-P ALL` options of the `mpstat` command to print on separate lines each CPU's statistics:

```
$ mpstat -P ALL
Linux 2.6.9-55.0.6.ELsmp (rmugprd.rmug.com)  07/04/2008
12:51:23 PM  CPU   %user   %nice %system %iowait    %irq   %soft   %idle    intr/s
12:51:23 PM  all   35.26    0.06    0.71    0.24    0.00    0.00   63.73   1008.94
12:51:23 PM    0   35.77    0.06    0.71    0.19    0.00    0.00   63.27    504.17
12:51:23 PM    1   34.74    0.07    0.72    0.29    0.00    0.00   64.18    504.77
```

The prior output shows that this server has two CPUs (indicated by a line for CPU 0 and a line for CPU 1). The `%idle` column is in the 60 percent range, indicating that there is some load on the CPUs on this box but not an inordinate amount. Table 8-5 describes the various statistics in the `mpstat` output.

Table 8-5. *Column Definitions for `mpstat` Processor Statistics*

Column	Description
`CPU`	Processor number. Starts at 0. The `all` row reports average statistics for all processors.
`%user`	Percentage of CPU utilization while executing at user level.
`%nice`	Percentage of CPU utilization while executing at user level with nice priority.
`%system`	Percentage of CPU utilization while executing at kernel level.

Table 8-5. *Column Definitions for* `mpstat` *Processor Statistics (Continued)*

Column	Description
`%iowait`	Percentage of time CPUs were idle during an outstanding disk I/O operation.
`%irq`	Percentage of time spent by CPUs servicing interrupts.
`%soft`	Percentage of time spent by CPUs to service software interrupts.
`%idle`	Percentage of time that CPUS were idle without outstanding disk I/O operations.
`intr/s`	Total number of interrupts received per second by CPU.

It's useful to compare the output of `mpstat` to that of `vmstat` (see recipe 8-1 for a discussion on using `vmstat`). Here you can confirm that the CPUs are 64 percent idle (`id` column) and there is no waiting on the I/O subsystem (`wa` column):

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff   cache  si   so    bi    bo   in    cs us sy id wa
 3  0  87164  40520   2312 1811872    0    0    79    65   19     6 35  1 64  0
```

You can also save the output of `mpstat` to a file. This next example saves to a file all CPU activity reported every 10 seconds for 100 times:

```
$ mpstat -P ALL 10 100 > mpperf.perf
```

This allows you to save performance statistics so that you can analyze and contrast performance for different time periods. See recipe 8-4 for a discussion on how to use the `sar` command to display the historical CPU usage.

8-4. Analyzing Historical CPU Load

Problem

You want to view the CPU load over the past several days.

Solution

The `sar` (system activity reporter) command is useful for displaying both current and historical processor load. Use `sar` with the `-u` option to report on CPU statistics. By default, `sar` will report on the current day's activities:

```
$ sar -u
```

To report on the previous day's worth of CPU statistics, use the `-f` option. The files that `sar` uses to report on statistics for different days of the month are located in the `/var/log/sa` directory and have the naming convention of `saNN`, where `NN` is the two-digit day of the month. For example, to have `sar` display CPU statistics for the tenth day of the month, run it as follows:

```
$ sar -u -f /var/log/sa/sa10
```

Here is a partial snapshot of the output:

```
02:40:01 PM       CPU     %user     %nice   %system   %iowait     %idle
02:50:01 PM       all      0.22      0.00      0.24      0.00     99.54
03:00:01 PM       all      0.22      0.00      0.24      0.00     95.53
03:10:01 PM       all      0.22      0.00      0.23      0.00     99.55
03:20:01 PM       all      0.42      0.00      1.06      2.11     96.41
03:30:01 PM       all      0.24      0.00      1.22      0.01     92.54
Average:          all      0.19      0.00      0.19      0.07     99.55
```

The columns in the prior output have the same meaning as the `mpstat` output and are described in Table 8-5. A low `%idle` could be an indication that the CPUs are underpowered or indicative of a high application load.

■ **Note** When you install the `sysstat` package, two `cron` jobs will be instantiated to create files used by the `sar` utility to report on historical server statistics. You can view these `cron` jobs by looking in the `/etc/cron.d/sysstat` file.

The `%iowait` column displays the time waiting for I/O. It follows that a high `%iowait` time indicates that the I/O subsystem is a potential bottleneck. See recipes 8-9 and 8-10 in this chapter for details on analyzing I/O performance.

After you identify a time in the past that had poor CPU performance, you can then run an Oracle Automatic Workload Repository (AWR) report for the same time period to correlate the operating system load to database activity. By default, Oracle will store seven days worth of AWR snapshots with 24 snapshots per day.

■ **Note** For complete details on using the AWR utility, see the Oracle Database Performance Tuning Guide. All of Oracle's documentation is available at `http://otn.oracle.com`.

MANUALLY RUNNING AWR, ADDM, AND ASH REPORTS

The AWR report is extremely useful for diagnosing database performance issues. To manually run an AWR report, log on to SQL*Plus as a privileged database account, and run the following script:

```
SQL> @?/rdbms/admin/awrrpt.sql
```

You will be prompted for input such as the type of report (HTML or text), the number of days, and the snapshot interval. The question mark (?) in the previous SQL statement is translated by SQL*Plus to the value of the `ORACLE_HOME` operating system variable.

The Automatic Database Diagnostic Monitor (ADDM) report is useful for tuning SQL statements. To manually run an ADDM report, log on to SQL*Plus as a privileged database schema, and run the following script:

```
SQL> @?/rdbms/admin/addmrpt.sql
```

Another useful report is the Active Session History (ASH) report. This report details recent active session activities. To run the ASH report, log into SQL*Plus as a privileged database schema, and run the following script:

```
SQL> @?/rdbms/admin/ashrpt.sql
```

See Chapter 11 for examples on how to automate the running of the previous reports for your environment.

How It Works

The `sar` utility is extremely useful because it allows you to analyze processor statistics for one of the three types of time periods:

- Real-time current statistics
- The current day's activities
- A previous day's activity

To use `sar` to report on real-time CPU statistics, specify a snapshot interval (in seconds) and the number of reports. The following displays current processor activity with a snapshot interval of 2 seconds for a total of 20 reports:

```
$ sar -u 2 20
```

To use `sar` to report on the current day's CPU activity, simply specify the `-u` option:

```
$ sar -u
```

To use `sar` to report on a previous day in the month, use the `-f` option. See the examples in the "Solution" section of this recipe for techniques for reporting on a previous day's statistics.

If you have multiple CPUs, you can view the output per CPU with the `-P ALL` options. You should now see one line per CPU in the output:

```
$ sar -u -P ALL
```

Here is a partial listing of the output:

```
04:30:01 PM       0      0.10      0.00      0.01      0.00     99.99
04:30:01 PM       1      0.11      0.00      0.01      0.00     99.98
```

8-5. Identifying Memory-Intensive Processes

Problem

You want to identify which Oracle session is consuming the most memory on the database server. If it's an Oracle session running a SQL query, then you want to display the associated SQL.

Solution

Use the `ps` command to identify the top memory-consuming Oracle processes and their associated process IDs:

```
$ ps -e -o pmem,pid,user,tty,args | grep -i oracle | sort -n -k 1 -r | head
```

Here is some sample output:

```
3.8   332 oracle ? oracleRMDB1 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
2.5 32092 oracle ? ora_mmon_RMDB1
2.4 32083 oracle ? ora_smon_RMDB1
1.6   329 oracle ? oracleRMDB1 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
1.5 32675 oracle ? oracleRMDB1 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
 1.3 32090 oracle   ?        ora_cjq0_RMDB1
```

From the second column in the previous output, the process with the ID of 332 is consuming 3.8 percent of the memory. Next you run the following query to identify Oracle-related details about process 332:

```
SET LINESIZE 80 HEADING OFF FEEDBACK OFF
SELECT
  RPAD('USERNAME : ' || s.username, 80) ||
  RPAD('OSUSER   : ' || s.osuser, 80) ||
  RPAD('PROGRAM  : ' || s.program, 80) ||
  RPAD('SPID     : ' || p.spid, 80) ||
  RPAD('SID      : ' || s.sid, 80) ||
  RPAD('SERIAL#  : ' || s.serial#, 80) ||
  RPAD('MACHINE  : ' || s.machine, 80) ||
  RPAD('TERMINAL : ' || s.terminal, 80) ||
  RPAD('SQL TEXT : ' || q.sql_text, 80)
FROM v$session s
    ,v$process p
    ,v$sql     q
WHERE s.paddr          = p.addr
AND   p.spid           = '&PID_FROM_OS'
AND   s.sql_address    = q.address(+)
AND   s.sql_hash_value = q.hash_value(+);
```

Here is the output for this example:

```
USERNAME : SYS
OSUSER   : oracle
PROGRAM  : rman@rmugprd.rmug.com (TNS V1-V3)
SPID     : 332
SID      : 538
SERIAL#  : 23
MACHINE  : rmugprd.rmug.com
TERMINAL :
SQL TEXT :
```

From the previous output, you can see that it is an RMAN backup job that is consuming the most memory resources. In this case, there is no associated SQL text with this job.

How It Works

The query in the "Solution" section of this recipe is a slight variation of the query presented in the "Solution" section of recipe 8-2. In this recipe, you outer join to the `V$SQL` view so the query will still return values even if there is no associated SQL query with the process being investigated.

This is useful for identifying `oracle` sessions that are consuming high amounts of system resources but are not related to a SQL query. These scenarios can occur if you're running Oracle utilities such as RMAN, Export/Import, Data Pump, and so forth.

8-6. Identifying Memory Bottlenecks

Problem

You want to view the current usage of memory on your database server.

Solution

Paging and swapping activity is an indicator of the efficiency of memory usage on your sever. In general, high amounts of paging and swapping indicate an inadequate amount of memory. Numerous utilities are available to monitor paging and swapping. For example, you can use `vmstat` (virtual memory statistics) to monitor the current memory usage. In this next line of code we generate `vmstat` reports every two seconds for a total of three reports:

```
$ vmstat 2 3
```

Here is some sample output:

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 2  0  80708  22224   2768 1886272    0    0   142    81    5     8 36  1 63  0
 2  0  80708  22800   2764 1885244    0    4  9356  3138 1120   190 84  3  0 13
 2  0  80708  23888   2680 1884288    0    0  9134    16 1103   217 84  3  0 14
```

If you have a fairly recent version of Linux, you can also use the `-a` option, which displays active and inactive memory. Here is an example of running `vmstat` with the `-a` option:

```
$ vmstat -a 2 3
```

Here's what the output looks like with the additional columns:

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free  inact active   si   so    bi    bo   in    cs us sy id wa
 2  0  85924  30992 849720 1147696    0    0   143    81    5     8 36  1 63  0
 2  0  85920  30864 849752 1147796    2    0  2828   480 1072   249 87  6  0  7
 2  0  85920  30032 849764 1148828    0    0     0 13600 1156    33 74  5  0 21
```

If your server shows high amounts of memory swapped in from disk (`si` column) or the amount of memory swapped out to disk (`so` column), then you may have a memory bottleneck. If you identify memory as being a bottleneck, refer to the "Solution" section of recipe 8-5 for identifying specific `oracle` processes consuming the most memory on your system.

How It Works

One of the main indicators of memory health is the amount of paging and swapping that is occurring. If you read five different Linux performance-tuning white papers, you'll get five slightly different definitions of paging and swapping. We're not going to split hairs about the exact definitions of those terms. We're going to state that "in general, paging and swapping are the movement of the contents of memory to and from disk."

Paging and swapping occur when there isn't enough physical memory to accommodate all the memory needs of the processes on the server. When paging and swapping take place, performance usually suffers. This is because the process of copying memory contents to and from disk is an inherently slow activity.

You can also use the `free` command to display current memory used, both physical and virtual (swap):

```
$ free
             total       used       free     shared    buffers     cached
Mem:       2057876    2040168      17708          0      55668    1805760
-/+ buffers/cache:     178740    1879136
Swap:      2031608        144    2031464
```

From the previous output, this system has 2GB of RAM of which almost all of it is used. It has about 2GB of swap space of which almost none is used. Don't be too alarmed if your Linux system is using most of its physical memory; that's typical on many Linux servers.

■ **Note** See Chapter 9 for details on using `ipcs` to view the memory and semaphores used by your database.

You can use the `-s` option to have the `free` command report output on a repeating interval. This example uses `free` to display memory usage in two-second snapshots and sends the output to a file:

```
$ free -s 2 > freemem.perf
```

Press Ctrl+C to exit from `free` when using the `-s` option. By default the `free` output reports memory usage in kilobytes. Use the `-m` to print in megabytes or the `-g` to display the output of `free` in gigabytes.

An effective way to use `free` is in combination with the `watch` command. The `watch` command is used to execute another program on a periodic basis. The next example uses `watch` to run the `free` utility every three seconds via the `-n` (interval) option. The `-d` (differences) option is used to have the output highlighted on the screen when there is a change in value from snapshot to snapshot:

```
$ watch -n 3 -d free

Every 3.0s: free                                Sun Aug  5 18:21:05 2007
             total       used       free     shared    buffers     cached
Mem:       2057876    2038240      19636          0      89840    1703248
-/+ buffers/cache:     245152    1812724
Swap:      2031608        144    2031464
```

You should be able visually see any changes in memory activity on your screen when running in this mode. To exit from `watch`, press Ctrl+C.

You can also view the current characteristics of memory by viewing the `/proc/meminfo` file. You can use the file to display the current physical memory and swap space being used. This example uses the `cat` utility to display the current memory usage:

```
$ watch -d cat /proc/meminfo
```

By default, the `watch` command will refresh the screen every two seconds. You should visually see differences highlighted from interval to interval:

```
MemTotal:      2074904 kB
MemFree:         23520 kB
Buffers:          2832 kB
Cached:        1838380 kB
SwapCached:        108 kB
Active:        1703208 kB
Inactive:       298916 kB
HighTotal:     1179584 kB
HighFree:         1024 kB
LowTotal:       895320 kB
LowFree:         22496 kB
SwapTotal:     4184924 kB
SwapFree:      4126964 kB
```

If you see an unusual amount of swap space being used (low `SwapFree`), then this is an indication that your server needs more memory. To exit from `watch`, press Ctrl+C.

8-7. Analyzing Historical Memory Load

Problem

You want to view the memory load for a previous day in the week.

Solution

Use `sar` with the `-f` (file) option to report on memory statistics for different days of the month. The files that `sar` uses to report statistics for different days of the month are located in the `/var/log/sa` directory and have the naming convention of `saNN`, where `NN` is the two-digit day of the month. For example, to have `sar` display memory paging statistics for the first day of the month, run it with the `-B` (report paging statistics) and `-f` (file) options as follows:

```
$ sar -B -f /var/log/sa/sa01
```

Here is a partial listing of the report:

```
11:10:01 AM  pgpgin/s pgpgout/s   fault/s  majflt/s
11:20:01 AM      0.02     16.17     18.37      0.00
11:30:01 AM      3.49     21.68     74.15      0.04
11:40:01 AM   4182.58    439.44    320.94      0.68
11:50:02 AM   4960.03   1027.79   4384.73      0.51
12:00:02 PM   4542.48   1156.96   6459.71      0.14
```

The previous output shows that around 11:40 a.m., there was a substantial increase in paging in from disk (`pgpgin/s`), pages paged out to disk (`pgpgout/s`), and page faults per second (`fault/s`).

Similarly, you can use the `-W` (report swapping statistics) option to view memory swapping:

```
$ sar -W -f /var/log/sa/sa01
```

Here is a partial snippet of the output:

```
11:10:01 AM  pswpin/s pswpout/s
11:20:01 AM      0.00      0.00
11:30:01 AM      0.01      0.00
11:40:01 AM      1.08      1.45
11:50:02 AM      0.81      2.97
12:00:02 PM      0.52      6.75
```

Unusually high values of pages swapped in to memory per second (`pswpin/s`) and pages swapped out per second (`pswpout/s`) are indications of inadequate memory. From the previous output, the system began to swap memory at around 11:40 a.m.

Because the `sar` utility reports on events that have happened in the past, you'll need to determine what system activity was taking place that caused any unusual spikes in memory usage.

After you identify a time in the past that had poor memory performance, you can then run an Oracle Automatic Workload Repository (AWR) report for the same time period to correlate the operating system load to database activity. By default, Oracle will store seven days' worth of AWR snapshots with 24 snapshots per day.

■ **Tip** For more details on using the AWR utility, see the Oracle Database Performance Tuning Guide. All of Oracle's documentation is available at `http://otn.oracle.com`.

How It Works

The `sar` utility is useful because you can use it to generate memory statistics in one of the following modes:

- Current real-time memory usage
- Current day's activity
- Previous day in the month

To report on real-time memory statistics, specify the `-W` option with an interval (in seconds) and the number of reports. This example generates current swapping statistics snapshots every three seconds for a total of ten reports:

```
$ sar -W 3 10
```

To report on the current day's worth of memory statistics, then do not provide `sar` with an interval or number of reports. The following example uses the `-B` option of `sar` to report on today's paging statistics:

```
$ sar -B
```

To report on a previous day's worth of memory statistics, use the `-f` option of `sar`. Refer to the "Solution" section of this recipe for examples on reporting on a previous day.

Several options are available with `sar` to report on memory. For example, the `-r` option will report extensive memory and swap utilization statistics:

```
$ sar -r
```

When run in this mode, the output can be wide and lengthy; it doesn't quite fit within the limits of this physical page. Refer to Table 8-6 for a description of each column.

```
02:40:01 PM kbmemfree kbmemused  %memused kbbuffers  kbcached kbswpfree kbswpused
%swpused  kbswpcad
02:50:01 PM     15460   2042416     99.25     64752   1654492   4031456       144
0.00         0
03:00:01 PM     15516   2042360     99.25     64772   1654472   4031456       144
0.00         0
```

Table 8-6. *Column Descriptions of the `sar -r` Output*

Column	Description
`kbmemfree`	Free memory in kilobytes
`kbmemused`	Amount of memory used in kilobytes
`%memused`	Percentage of used memory
`kbbuffers`	Amount of memory used as buffers in kilobytes
`kbcached`	Amount of memory used to cache data by the kernel in kilobytes
`kbswpfree`	Amount of free swap space in kilobytes
`kbswpused`	Amount of used swap space in kilobytes
`%swpused`	Percentage of used swap space
`kbswpcad`	Amount of cached swap memory in kilobytes

8-8. Monitoring Disk Space

Problem

You want to proactively monitor disk space so that you're not surprised in the middle of the night by a disk becoming full.

Solution

Use a shell script like the one listed next to proactively monitor disk space:

```
#!/bin/bash
#
inCheck='/dev/sda1 /dev/sda2'
for mntList in $inCheck
do
usedSpc=$(df -h $mntList|awk '{print $5}'|egrep -iv 'Use|Capacity'| \
cut -d "%" -f1 -)
#
case $usedSpc in
[0-9])
  diskStat="relax, lots of disk space: $usedSpc"
;;
[1-7][0-9])
  diskStat="disk space okay: $usedSpc"
;;
[8][0-9])
  diskStat="WARNING, disk space low: $mntList $usedSpc percent full"
;;
[9][0-9])
  diskStat="ALERT, running out of space: $mntList $usedSpc percent full"
;;
[1][0][0])
  diskStat="ERROR, no space left: $mntList $usedSpc percent full"
;;
*)
diskStat="huh?: $usedSpc"
esac
#
BOX=$(uname -a | awk '{print $2}')
sLine="Disk space issue on: $BOX, mount point $mntList"
echo $diskStat|egrep 'ALERT|ERROR' && echo $diskStat|mailx -s "$sLine" prd@spt.com
done
#
exit 0
```

You'll have to modify variables in the previous script to match your environment. For example, the `inCheck` variable should hold the mount points you are interested in monitoring.

Refer to Chapter 11 for details on how to automate the running of a script. See Chapter 7 for techniques used to create a shell script.

How It Works

The `df` (disk free) command is used frequently by DBAs to determine the amount of disk-free space on a server. The default output of `df` lists a disk's used and available space in kilobytes for all mounted filesystems:

```
$ df

Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      74699952   9304680  61600740  14% /
/dev/sda1               101086     14092     81775  15% /boot
none                   1028936         0   1028936   0% /dev/shm
```

The previous output can sometimes be difficult to read. Use the `-h` (human-readable) option of `df` to have the free space automatically displayed in kilobytes, megabytes, gigabytes, or terabytes. The `df -h` output will adjust the space amount abbreviation (K, M, G, T) depending on the size of each filesystem:

```
$ df -h

Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                       72G  8.9G   59G  14% /
/dev/sda1              99M   14M   80M  15% /boot
none                 1005M     0 1005M   0% /dev/shm
```

You can also use `df` to view disk space on a particular mount point. For example, to check for free space in the `/tmp` mount point, use this:

```
$ df -h /tmp

Filesystem             size   used  avail capacity  Mounted on
swap                    15G   2.3M    15G     1%    /tmp
```

The `du` (disk usage) command is another extremely useful utility for monitoring disk space. When a filesystem fills up, the DBA needs to quickly determine what directories are using up what space. The `du` command helps you quickly determine where the disk space is being consumed.

By default, `du` will recursively display disk usage for each directory below a parent directory as well as show an aggregate amount of space usage of all directories beneath a parent. For example, to display the disk space usage of all directories below the `/ora01` mount point, use `du`, as shown here:

```
$ du /ora01
```

Here's a partial listing of the output:

```
848     /ora01/10g/database/install
276     /ora01/10g/database/response
801668  /ora01/10g/database
1585268 /ora01/10g
12      /ora01/oraInventory/locks
40      /ora01/oraInventory/logs/results/db
```

The default output of `du` is displayed in kilobytes. Use the `-h` switch to make the output more readable and also display an appropriate space amount abbreviation (K, M, G, or T):

```
$ du -h /ora01
848K    /ora01/10g/database/install
276K    /ora01/10g/database/response
783M    /ora01/10g/database
1.6G    /ora01/10g
12K     /ora01/oraInventory/locks
40K     /ora01/oraInventory/logs/results/db
```

To report an aggregated disk space amount used in a directory and its subdirectories, use the `-s` option. This example uses `-s` and `-h` to print an aggregated total in human-readable form:

```
$ du -sh /ora01
4.6G    /ora01
```

Another clever way DBAs use `du` is to display the top directories and files that consume space below a given parent directory. This command will display the top five directories in terms of disk space used below a given directory:

```
$ du -s * | sort -nr | head -5
```

The previous command would get a little tedious to type in over and over again. To resolve this, create an alias that points to the command:

```
$ alias tn='du -s * | sort -nr | head -5'
```

An alternate technique to creating an alias is to create a function (see Chapter 4 for details). This next bit of code creates a Bash shell function to allow you to dynamically specify the top number of disk usage directories to be displayed. If you're not using the Bash shell, type the command `bash` to set your shell appropriately:

```
$ bash
```

Then create the function by typing the following lines:

```
$ function tnf {
> du -s * | sort -nr | head -$1
> }
```

The `$1` variable holds the first parameter passed into the `tnf` function. You can call the `tnf` function directly from the command line and pass it in the number of top directories you want to view (in this example ten):

```
$ tnf 10
```

8-9. Monitoring I/O

Problem

You want to determine whether your disk storage is a bottleneck.

Solution

The `iostat` command can help you determine whether disk I/O is potentially a source of performance problems. The `-x` (extended) option used with the `-d` (device) option is a useful way to generate I/O statistics. This next example uses the `-x` and `-d` options to display extended device statistics every ten seconds:

```
$ iostat -xd 10
```

You need a really wide screen to view this output; here's a partial listing:

```
Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
sda          0.01   3.31  0.11  0.31    5.32   28.97     2.66    14.49    83.13
0.06  138.44   1.89   0.08
```

■ **Note** On some Linux/Unix distributions, the `iostat` output may report the disk utilization as %b (percent busy).

This periodic extended output allows you to view in real time which devices are experiencing spikes in read and write activity. To exit from the previous `iostat` command, press Ctrl+C. Table 8-7 describes the I/O-related columns in the `iostat` output.

When trying to determine whether device I/O is the bottleneck, here are some general guidelines when examining the `iostat` output:

- Look for devices with abnormally high blocks read or written per second.
- If any device is near 100 percent utilization, that's a strong indicator I/O is a bottleneck.

If the bottlenecked disks are used by Oracle, then you can query the data dictionary to identify sessions with high I/O activity. The following query is useful for determining which SQL statements generate the most read/write activity:

```
SELECT *
FROM
(SELECT
  parsing_schema_name
 ,direct_writes
 ,SUBSTR(sql_text,1,75)
 ,disk_reads
FROM v$sql
ORDER BY disk_reads DESC)
WHERE rownum < 20;
```

The next query is useful for determining which objects produce the heaviest I/O activity in the database:

```
SELECT *
FROM
(SELECT
  s.statistic_name
 ,s.owner
 ,s.object_type
 ,s.object_name
 ,s.value
  FROM v$segment_statistics s
  WHERE s.statistic_name IN
    ('physical reads', 'physical writes', 'logical reads',
     'physical reads direct', 'physical writes direct')
ORDER BY s.value DESC)
WHERE rownum < 20;
```

How It Works

If you execute `iostat` without any options, then you'll get a default report that displays averages since the system was last started:

```
$ iostat
avg-cpu:  %user   %nice    %sys %iowait   %idle
          18.91    0.04    1.20    0.15   79.70

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               7.14       398.01       409.52  164484368  169239542
sda1              0.00         0.00         0.00       1538        166
sda2             54.15       396.92       407.74  164032098  168505032
sda3              0.35         1.04         1.77     429820     733168
```

Notice that there are two sections in the prior `iostat` output. The first section is the CPU Utilization Report. The second section relates to disk I/O and is referred to as the Device Utilization Report. Table 8-7 describes the columns used for disk I/O. Use the `-d` option of `iostat` to display only device statistics.

Table 8-7. *Column Descriptions of* `iostat` *Disk I/O Output*

Column	Description
Device	Device or partition name.
`tps`	I/O transfers per second to the device.
`Blk_read/s`	Blocks per second read from the device.
`Blk_wrtn/s`	Blocks written per second to the device.
`Blk_read`	Number of blocks read.
`Blk_wrtn`	Number of blocks written.
`rrqm/s`	Number of read requests merged per second that were queued to device.
`wrqm/s`	Number of write requests merged per second that were queued to device.
`r/s`	Read requests per second.
`w/s`	Write requests per second.
`rsec/s`	Sectors read per second.
`wsec/s`	Sectors written per second.
`rkB/s`	Kilobytes read per second.
`wkB/s`	Kilobytes written per second.
`avgrq-sz`	Average size of requests in sectors.
`avgqu-sz`	Average queue length of requests.
`await`	Average time in milliseconds for I/O requests sent to the device to be served.
`svctm`	Average service time in milliseconds.
`%util`	Percentage of CPU time during which I/O requests were issued to the device. Near 100 percent indicates device saturation.

You can also instruct `iostat` to display reports at a specified interval. The first report displayed will report averages since the last server reboot; each subsequent reports shows statistics since the previously generated snapshot. The following example displays a device statistic report every three seconds:

```
$ iostat -d 3
```

To exit from the previous `iostat` command, press Ctrl+C. You can also specify a finite number of reports that you want generated. This is useful for gathering metrics to be analyzed over a period of time. This example instructs `iostat` to report every 2 seconds for a total of 15 reports:

```
$ iostat 2 15
```

When working with locally attached disks, the output of the `iostat` command will clearly show where the I/O is occurring. However, it is not that clear-cut in environments that use external arrays for storage. What you are presented with at the filesystem layer is some sort of a

virtual disk that might also have been configured by a volume manager. Virtual disks are often referred to as *volumes* or *logical units* (LUNs). A LUN is a logical disk that physically comprises one or more physical disks. The LUN represents the virtualization layer between the physical disks and the applications running on the database server. Figure 8-1 illustrates at a high level the abstraction involved with virtual disks.

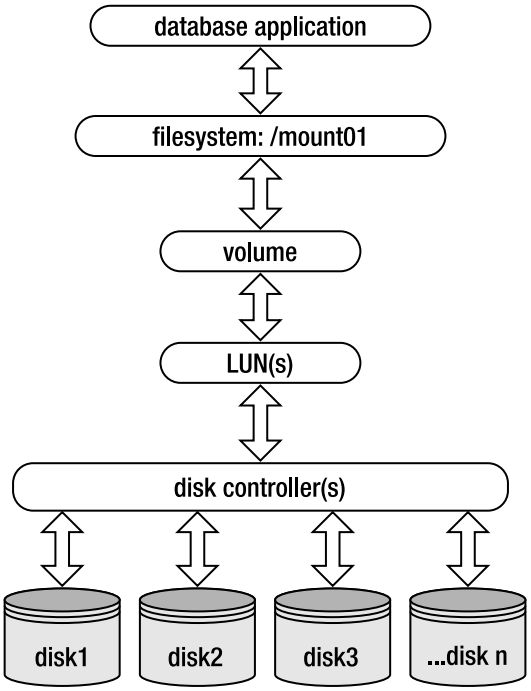

Figure 8-1. *Abstraction layers between database application and physical disks*

When working with virtual disks, the output from `iostat` will report on read/write activity at the virtual disk level, not the underlying physical disks. In these situations, there may be many layers of abstraction between the database application and physical disks. This can make it difficult to isolate the exact source of an I/O bottleneck. We recommend you work closely with your storage system administrator to determine whether a particular set of LUNs (and underlying physical disks) are a source of poor I/O performance.

WHAT'S WRONG WITH THE DATABASE?

One of us was recently involved with a performance crisis with a production database system. A materialized view refresh job that used to take ten minutes was now taking four hours. The managers and system administrators were asking the DBAs for answers.

One of the DBAs inspected the output from `iostat` and noticed the utilization (`%util`) for several disks was near 90 percent. Next, the DBA used the `dd` command to generate some large files on the Oracle filesystem to get timings on a file creation:

```
$ time dd if=<database filename> of=<test filename>
```

The DBA performed this test on the production server and then executed the same test on a test box with a similar disk layout (as production). The time to create a large file was five times faster on the test box. This allowed the DBA to show the system administrators that something was wrong with I/O on the production box. The SAs reconfigured the disks in production, and the MV refresh job went back down to ten minutes.

8-10. Analyzing Historical I/O Load

Problem

You want to view disk I/O activity for the past several days.

Solution

Use the `sar` utility with the `-f` (file) option to report on statistics for different days of the month. This is a useful tuning and troubleshooting feature because it allows you to analyze metrics over a period of several days.

To report on historical statistics, the `sar` command uses files located in the `/var/log/sa` directory. These files have the naming convention of `saNN`, where `NN` is the two-digit day of the month. For example, to have `sar` display disk statistics for the tenth day of the month, run it with the `-d` and `-f` options as follows:

```
$ sar -d -f /var/log/sa/sa10
```

Here's a partial snippet of the output (this output may vary depending on your version of Linux and the `sar` command):

```
02:40:01 PM    DEV              tps    rd_sec/s  wr_sec/s
03:50:01 PM    dev1-14          0.00       0.00      0.00
03:50:01 PM    dev1-15          0.00       0.00      0.00
03:50:01 PM    dev8-0          30.02     642.39   2824.27
03:50:01 PM    dev9-0           0.00       0.00      0.00
```

The `tps` column shows the I/O operatings (transfers) per second to the device. The `rd_sec/s` is the number of sectors read from the device. The `wr_sec/s` is the number of sectors written to the device. Unusually high read and write rates indicate an overworked disk subsystem.

At the bottom of the `sar` report, the averages over the period of reporting time are displayed:

```
Average:       dev1-14          0.00       0.00      0.00
Average:       dev1-15          0.00       0.00      0.00
Average:       dev8-0          31.68    2331.40   1978.12
Average:       dev9-0           0.00       0.00      0.00
```

After you identify a time in the past that had a heavy I/O load, you can then run an Oracle AWR report for the same time period to correlate the operating system load to database activity. By default, Oracle will store 7 days' worth of AWR snapshots with 24 snapshots per day.

■ **Tip** For more details on using the AWR utility, see the Oracle Database Performance Tuning Guide. All of Oracle's documentation is available at `http://otn.oracle.com`.

How It Works

The `sar` utility is powerful because you can use it to generate device I/O statistics in one of the following types of output:

- Current real-time memory usage
- Current day's activity
- Previous day of the month statistics

To report current real-time I/O statistics, use the `-d` option with an interval (in seconds) and the number of reports you want generated. The following syntax instructs `sar` to report disk statistics every 2 seconds for a total of 12 reports:

```
$ sar -d 2 12
```

While reporting on real-time statistics, use the `-o` (out) option to send output to a file:

```
$ sar -d 2 12 -o sarout.perf
```

This creates a binary output file that can later be used to analyze disk I/O metrics. At some later point you can use `sar` with the `-f` option to report on the contents of that file.

To report on the current day's worth of activity, specify the `-d` option with no time interval:

```
$ sar -d
```

To report device I/O statistics on a previous day of the month, see the "Solution" section of this recipe for an example.

8-11. Monitoring Network Traffic

Problem

You suspect that the network may be a bottleneck. You want to view network statistics.

Solution

Use the `netstat` (network statistics) command to display network traffic. Perhaps the most useful way to view `netstat` output is with the `-ptc` options. These options display the process ID and TCP connections, and they continuously update the output:

```
$ netstat -ptc
```

Press Ctrl+C to exit the previous command. Here's a partial listing of the output:

```
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address  Foreign Address  State       PID/Program name
tcp        0      0 rmug.com:62386 rmug.com:1521    ESTABLISHED 22864/ora_pmon_RMDB
tcp        0      0 rmug.com:53930 rmug.com:1521    ESTABLISHED 6091/sqlplus
tcp        0      0 rmug.com:1521  rmug.com:53930   ESTABLISHED 6093/oracleRMDB1
tcp        0      0 rmug.com:1521  rmug.com:62386   ESTABLISHED 10718/tnslsnr
```

If the `Send-Q` (bytes not acknowledged by remote host) column has an unusually high value for a process, this may indicate an overloaded network. The useful aspect about the previous output is that you can determine the operating system process ID (PID) associated with a network connection. If you suspect the connection in question is an `oracle` session, you can use the techniques described in the "Solution" section of recipe 8-2 to map an operating system PID to an `oracle` process or SQL statement.

■ **Note** The `/proc/net` directory stores information about current network settings and activity.

How It Works

When experiencing performance issues, usually the network is not the cause. Most likely you'll determine that bad performance is related to a poorly constructed SQL statement, inadequate disk I/O, or not enough CPU or memory resources. However, as a DBA, you need to be aware of all sources of performance bottlenecks and how to diagnose them. In today's highly interconnected world, you must possess network troubleshooting and monitoring skills. The `netstat` utility is a good starting place for monitoring server network connections.

You can also use the `sar` command with the `-n` option to report on network statistics. The `-n` option takes as an argument one of the following: `DEV` (network devices), `EDEV` (error count), `SOCK` (sockets), or `FULL` (all). The following command displays the current day's network device statistics:

```
$ sar -n DEV
```

Here's a limited listing of the output:

```
12:00:01 AM  IFACE  rxpck/s txpck/s  rxbyt/s  txbyt/s  rxcmp/s  txcmp/s  rxmcst/s
12:10:01 AM     lo     0.00    0.00     0.00     0.00     0.00     0.00      0.00
12:10:01 AM   eth0     0.34    0.11    39.17    10.22     0.00     0.00      0.04
12:10:01 AM   eth1     0.00    0.00     0.00     0.00     0.00     0.00      0.00
12:10:01 AM   sit0     0.00    0.00     0.00     0.00     0.00     0.00      0.00
```

The previous output shows the number of packets transmitted and received per second, as well as the bytes and compressed packets. The `sar -n` output allows you to examine the current day's network traffic on snapshots taken on ten-minute intervals.

TROUBLESHOOTING DATABASE NETWORK CONNECTIVTY

Use these steps as guidelines when diagnosing Oracle database network connectivity issues:

1. Use `ping` to determine whether the remote box is accessible. If `ping` doesn't work, work with your system or network administrator to ensure you have server-to-server connectivity in place.
2. Use `tnsping` to determine whether Oracle Net is working. This utility will verify that an Oracle Net connection can be made to a database via the network. If `tnsping` can't contact the remote database, verify that the remote listener and database are both up and running. On the remote box, use the `lsnrctl status` command to verify that the listener is up. Verify that the remote database is available by establishing a local connection as a non-`SYS` account (`SYS` can often connect to a troubled database when other schemas will not work).
3. Verify that the `tns` information is correct. If the remote listener and database are working, then ensure that the mechanism for determining `tns` information (like the `tnsnames.ora` file) contains the correct information. Sometimes the client machine will have multiple `TNS_ADMIN` locations and `tnsnames.ora` files. One way to verify whether a particular `tnsnames.ora` file is being used is to rename it and see whether you get a different error when attempting to connect to the remote database.

If you're still having issues, examine the client `sqlnet.log` file and the remote server `listener.log` file. Sometimes these files will show additional information that will pinpoint the issue.