COMPAQ

Performance Tuning for Oracle7 on Compaq Systems Running SCO UNIX

Compaq TechNote

Includes information on:

- General performance-tuning guidelines
- Tuning methdology

NOTICE

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

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

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

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

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

Performance Tuning for Oracle7 on Compaq Systems Running SCO UNIX

First Edition (July 1993)
Part Number 145942-001

Contents

Chapter 3
Tuning the System Processor and Reducing Oracle Contention

Chapter 4
Performance Analysis Tools Reference

Chapter 1
Introduction

This COMPAQ TechNote is written for systems integrators or administrators with experience in configuring a system using ORACLE7 and a COMPAQ EISA-based product.

In addition, this TechNote presumes that you have ORACLE7 and SCO UNIX version 3.2.4.2, or later, installed and operating.

TechNote Contents

This TechNote contains general performance-tuning guidelines for ORACLE7. These guidelines are not the only steps or approaches you could take to improve system performance. For an in-depth discussion of performance tuning considerations, see the following Oracle documents:

- *ORACLE RDBMS Performance Tuning Guide Version 7.0*
- *ORACLE for UNIX Performance Tuning Tips*

You can also find information on the SQL*DBA parameter files and the data dictionary in the Oracle document, *ORACLE RDBMS Database Administrator's Guide Volume III Version 7.0.* For information on installing ORACLE7, see the Oracle document, *ORACLE for UNIX Installation and Configuration Guide.*

This TechNote supplements the Oracle documents shown above with tuning considerations specific to the COMPAQ SYSTEMPRO/XL. Use this TechNote as a "field guide" to performance tuning and the Oracle documents as an
in-depth reference. For information on tuning the SCO UNIX operating system on a COMPAQ EISA-based product, see the COMPAQ TechNote: *Performance Tuning for SCO UNIX on COMPAQ Systems.*

Use the following table to locate the information you want in this TechNote:

Table 1-1
TechNote Chapter Summary

Chapter 2 TUNING MEMORY AND DISK I/O	Provides a recommended approach to identify memory and disk I/O problems. Includes suggestions for tuning.
Chapter 3 TUNING THE SYSTEM PROCESSOR AND REDUCING ORACLE CONTENTION	Provides a recommended approach to identify system processor and ORACLE contention problems. Includes suggestions for tuning.
Chapter 4 PERFORMANCE ANALYSIS TOOLS REFERENCE	Provides reference information on the UNIX performance analysis tools used in this TechNote.

Tuning Methodology

The first activity in the tuning process is to identify which system areas require additional tuning: a hardware subsystem, the UNIX operating system, or ORACLE7. If ORACLE7 requires additional tuning, you can use UNIX and ORACLE7 performance analysis tools to isolate and tune the appropriate areas.

Performance tuning is a step-by-step refinement process that you might need to repeat. Use the tuning methodology steps in this TechNote to identify performance bottlenecks and make the recommended changes. You can then repeat these steps, if necessary, to further improve performance.

The tuning methodology described in this TechNote focuses on tuning ORACLE7. For information on tuning the SCO UNIX operating system, see the COMPAQ TechNote, *Performance Tuning for SCO UNIX on COMPAQ Systems.*

The four steps in the tuning methodology are:

1. Tuning applications and SQL statements
2. Tuning memory
3. Tuning disk I/O
4. Tuning the system processor and reducing contention

The order of these tuning steps is important. We list steps that are likely to provide the greatest performance gain first.

Database and applications design and implementation are also important performance considerations. A poorly designed and coded database application might perform poorly, even with a well-planned, tuned installation.

Ideally, your database and applications design tuning should be done *before* you install the database. Once the database is operating, it may be difficult and time-consuming to correct design problems.

Relational database design methodology and SQL tuning are beyond the scope of this TechNote. For in-depth discussions on using SQL-level tuning facilities, see the *ORACLE RDBMS Database Administrator's Guide Version 7.0*, and Chapter 2, "Tuning SQL Statements," of the *ORACLE RDBMS Performance Tuning Guide Version 7.0*. Refer to these chapters and other relevant Oracle application product guides for further information.

The focus of this TechNote is on tuning memory, disk I/O, the system processor, and reducing ORACLE contention. We recommend using UNIX performance analysis tools to identify which subsystem, if any, might be adversely impacting performance. Once you identify the subsystem, we recommend tuning it with the **monitor** utility and other ORACLE tools.

Performance Monitoring Tools

Performance tuning requires two levels of monitoring tools. The first level monitors and analyzes overall system activity, including hardware and the operating system supporting ORACLE and its applications. The second level monitors and analyzes ORACLE internal activity.

UNIX Performance Analysis Tools

Use the UNIX performance analysis tools to identify which, if any, system component might be adversely affecting performance and to determine how tuning could improve ORACLE performance. If related to ORACLE and its applications, your next step is to identify the responsible elements within the database, using the ORACLE monitoring tools.

For this TechNote, we used the UNIX **sar** utility to identify system component-level performance bottlenecks. Other third-party utilities perform more functions and are easier to use, but **sar** provides the necessary data to monitor system component-level performance, and it is widely available.

See Chapter 4, "Performance Analysis Tools Reference" for details on the **sar** command.

ORACLE Performance Monitoring Tools

The ORACLE SQL*DBA **monitor** utility monitors database activity, behavior, and statistics so you can tune the database for improved performance. See Chapter 4, "Performance Analysis Tools Reference" for information on the **monitor** utility.

For detailed **monitor** utility information, see Appendix B, "The SQL*DBA Reference," and Chapter D, "The SQL*DBA Monitors," of the *ORACLE RDBMS Database Administrator's Guide Volume III Version 7.0.*

To create the views and public synonyms for the dynamic performance tables, run the **monitor** utility. To grant monitor privileges, run **catalog.sql** and **monitor.sql**. These files are in the *$ORACLE_HOME/rdbms/admin* directory.

You can invoke the **monitor** utility after you run SQL*DBA. To run **monitor**, follow these steps:

1. At the SQLDBA prompt, enter:

   ```
   connect internal
   ```

   The system displays a menu of all the monitor options and parameters.

2. To access the top menu line, press **F5**.

3. Scroll down to the option you want, and press **ENTER**.

   The system displays the Monitor Information window for the option you selected.

4. To return to the SQLDBA Monitor Menu, press the **CTRL+C** keys.

   NOTE: You may also enter the monitor options and parameters directly in the command line. For example, at the SQLDBA prompt, enter:

   ```
   monitor stat cache 0
   ```

   Where: stat specifies the statistics option

   cache specifies the cache statistics

   0 (zero) is the ORACLE-assigned process ID (PID)

   A value of 0 specifies system-wide statistics. If you are interested in a specific ORACLE process, you must first find the PID by using the **process** option.

Chapter 2
Tuning Memory and Disk I/O

Database applications are typically characterized as both memory and I/O intensive. Disk subsystem performance is the limiting factor. Insufficient memory can also contribute to compromised database performance. ORACLE7, the user and shadow processes, and the ORACLE System Global Area (SGA) require ample amounts of memory, as dictated by applications and concurrent use.

Because memory access is much faster than disk access, a tuning goal is to satisfy data requests from memory rather than from the disk. To do this, allocate as much memory as possible to the SGA so that the database can cache as much data in memory as possible. However, do not allocate so much memory to the SGA that it causes the UNIX operating system to require swapping.

Without sufficient memory, the operating system performs paging and swapping of both system and ORACLE processes or SGA segments. The ORACLE and non-ORACLE processes then compete for system processor resources.

Identifying Memory Problems

Increased disk I/O occurs when a system short on memory performs paging and swapping to and from the swap device. Disk I/O contention is more severe if the Program Global Area (PGA) segments are also paged out. The processing overhead of paging and swapping increases the system processor load. This can aggravate resource contention.

To determine if insufficient memory is adversely affecting performance, collect and analyze system activity information. To collect this information, use the UNIX **sar** command with the **-rwp** options. After collecting the data, analyze the **-rwp** options report. For detailed information on the **sar** command, see Chapter 4, "Performance Analysis Tools Reference."

Tuning Overall System Memory

If you determine that excessive paging and swapping is occurring, tune the overall system memory performance. For tuning guidelines, see the COMPAQ TechNote, *Performance Tuning for SCO UNIX on COMPAQ Systems.* After tuning the overall system memory subsystem, you can tune the ORACLE memory allocations.

Tuning memory involves optimizing the use of UNIX and ORACLE buffer caches and eliminating unnecessary buffers. However, you might have to trade between memory and disk I/O performance. In that case, consider adding more physical memory to your system.

Tuning the UNIX Buffer Cache

The UNIX buffer cache resides in system space. This cache is a pool of memory pages that are used to buffer data from and to disk. Unless specified, the amount of memory allocated to the buffer pool is a percentage of the total memory. This might be more than necessary. Any reduction to the buffer cache increases the available memory to the user processes. The following figure illustrates the memory components of a UNIX system running ORACLE7.

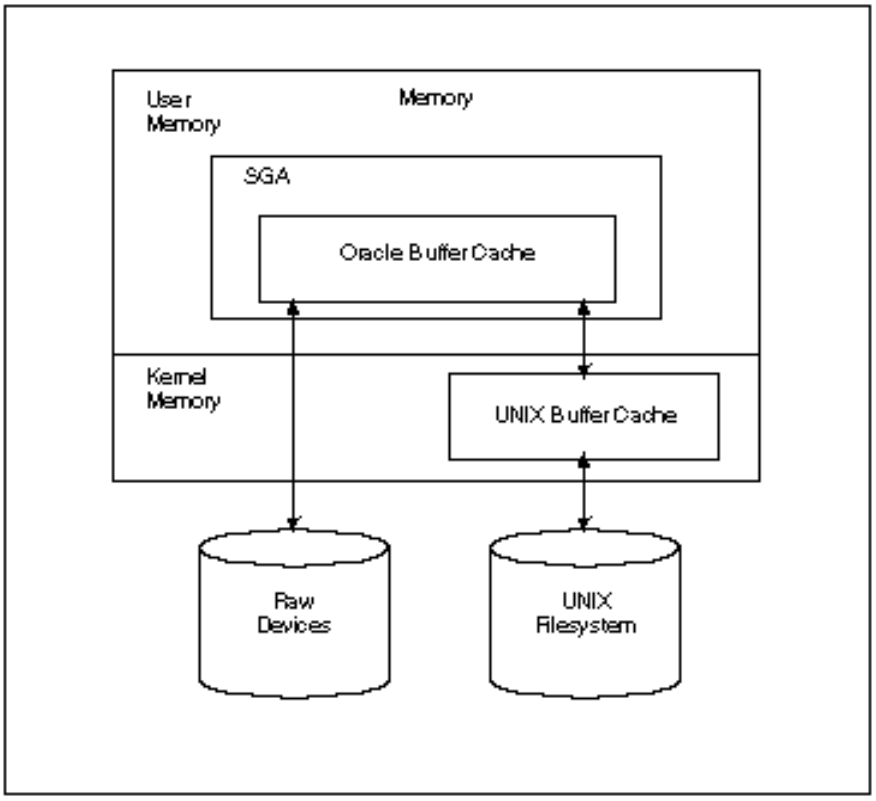

Figure 2-1. UNIX and ORACLE Buffer Cache

For recommendations on tuning the UNIX buffer cache, see the COMPAQ TechNote, *Performance Tuning for SCO UNIX on COMPAQ Systems*.

Tuning the ORACLE SGA Allocation

In addition to the UNIX buffer cache, another significant memory area that you can tune is the SGA. The SGA is a memory structure for one ORACLE database instance. SGA is allocated in the UNIX shared memory space when the ORACLE instance starts.

Data in the SGA is kept in caches. A cache is a memory location that holds copies of data on its way to or from the disk. Memory reads and writes can access the data stored in a cache more quickly than through disk I/O.

Because the purpose of the SGA is to store data in memory for fast access, try to keep the SGA in main memory. When the SGA swaps to disk, its data is no longer so quickly accessible.

You can make ORACLE read the entire SGA into memory when you start your instance by setting the value of the initialization parameter **pre_page_sga** to YES. This setting might increase the amount of time needed for instance startup, but it is likely to decrease the time needed for ORACLE to reach its full performance capacity after startup. This setting does *not* prevent the operating system from paging or swapping the SGA after it is initially read into memory.

The SGA contains the following tunable caches:

- ORACLE buffer cache
- Redo log buffer
- Data dictionary cache
- Rollback and temporary segment blocks

Tuning the ORACLE Buffer Cache

The ORACLE buffer cache contains copies of recently modified data blocks. Each buffer corresponds to a modified database block. The ORACLE buffer manager places frequently accessed data in the buffer cache, thereby reducing the need for disk I/O requests. These data blocks contain data for:

- tables
- indexes
- rollback segments
- clusters

The initialization parameter **db_block_buffers** determines the *number* of buffers in the ORACLE buffer cache. The parameter **db_block_size** determines the *size* of each buffer. The default values for these parameters depend on the operating system. For more information on initialization parameters, see Appendix E, "Parameter Files and Initialization Parameters" in the Oracle document *ORACLE RDBMS Database Administrator's Guide Volume III Version 7.0.*

The optimal ORACLE buffer cache size depends on the overall system load and on the relative priority of ORACLE over other applications. Increasing the ORACLE buffer cache size might affect other applications by reducing the amount of available memory.

The relative size of the UNIX and ORACLE buffer caches is also important. Both caches share the same limited memory resources. When you use a UNIX filesystem, UNIX first reads data from disk into the UNIX buffer cache, then from the UNIX buffer cache to the ORACLE buffer cache.

If you use raw devices, you can bypass the UNIX buffer cache (see Figure 2-1). Depending on the amount of non-ORACLE process disk I/O, you might then be able to reduce the size of the UNIX buffer cache without affecting performance.

Using raw devices can improve the performance of an I/O-bound system, but this improvement comes with significant consequences. Raw devices are not as flexible as filesystems. Raw devices complicate the process to monitor and maintain partitions and administer tablespace growth. In addition, you can only back up raw devices by using the **dd** utility. This makes backups more cumbersome.

Optimize Number of Redo Buffers

The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is used for database recovery.

You can monitor the redo log space requests statistic by using the **monitor** command. This command can produce a statistic that indicates the number of times a user process waits for space in the redo log buffer.

1. At the SQLDBA prompt, enter:

   ```
   monitor sessionstatistic
   ```

   The ORACLE Session Statistics Monitor screen displays.

2. Use the **TAB** key and press **F11** key to select "Redo."

   A display similar to the following appears:

ORACLE Session Statistics Monitor

Starting Session ID: Ending Session ID:

[ ] User [X] Redo [ ] Enqueue [ ] Cache [ ] ParServer [ ] SQL

Statistic Name	Current	Average	Minimum	Maximum	Total
redo blocks written	0	0	0	0	13
redo buffer allocation entries	0	0	0	0	0
redo entries	0	0	0	0	52
redo entries linearized	0	0	0	0	0
redo log space requests	0	0	0	0	0
redo log space wait time	0	0	0	0	0
redo log switch interrupts	0	0	0	0	0
redo ordering names	0	0	0	0	0
redo size	0	0	0	0	2586

The value of *redo log space requests* should always be 0. A non-zero value indicates that processes are waiting for space in the buffer. To correct this, increase the size of the redo log buffer by modifying the parameter **log_buffer**.

Reduce Redo Log Buffer Latch Contention

Latches regulate access to the redo log buffer. Heavy access to the redo log buffer can cause contention for these latches. Examine the activity of the redo log buffer latches by using the **monitor** command.

At the SQLDBA prompt, enter:

```
monitor latch
```

This command displays latch activity similar to the following:

ORACLE Latch Monitor

Latch Name	Holder PID	Willing-to-Wait-Requests Gets	Misses	Sleeps	No-Wait-Requests Gets	Misses
process allocation						
session switching						
•						
•						
•						
redo allocation						
messages						
•						
•						
•						
redo copy						

Use the **TAB** key to locate the cursor in the "Latch Name" column, then use the up and down arrow keys to locate the *redo allocation* and *redo copy* latches.

Examine the statistics for the *redo allocation* and *redo copy* under the column, "Willing-to-Wait-Requests." "Total" = total number of requests for the latch. "Timeouts" = number of timeouts while waiting for the latch.

If the ratio of Timeouts to Total for a particular latch exceeds 10 or 15 percent, contention for that latch might be affecting performance. Some systems with multiple system processors can tolerate more contention without performance reduction. For more specific information, see the *ORACLE for UNIX Installation and Configuration Guide*.

To reduce contention for the redo allocation latch, minimize the time that any single process holds the latch by decreasing the value of the parameter **log_small_entry_mas_size**.

To reduce contention for the redo copy latches in multiprocessor environments, perform either of the following actions:

- Increase the value of the parameter **log_simultaneous_copies**.
- Reduce the time each process holds a latch through forcing the ORACLE user process to "pre-build" redo entries before obtaining a redo copy latch.

For more information, see the *ORACLE RDBMS Performance Tuning Guide Version 7.0.*

Optimize Data Dictionary Cache

The data dictionary is a collection of database tables and views containing reference information about the database, its structures, and its users. The data dictionary stores:

- Names of all tables and views in the database.
- Names and datatypes of columns in database tables.
- Privileges of all ORACLE users.

This information is useful as reference material for database administrators, application designers, and end users. ORACLE also frequently accesses the data dictionary during the parsing of SQL statements. This access is essential to the continuing operation of ORACLE.

Because ORACLE accesses the data dictionary often, a special location in memory holds the dictionary data. All ORACLE user processes share this area.

The initialization parameter **shared_pool_size** determines the memory allocation for the data dictionary cache. This parameter specifies the amount of space in the SGA allocated to the shared pool. The default value for this parameter is 3.5 megabytes. Because the data dictionary cache is contained in the shared pool, this parameter specifies the amount available to the library cache.

For best performance when parsing SQL statements, the data dictionary cache must be large enough to hold all the most-often accessed dictionary data. Data dictionary cache misses generate "recursive calls." Recursive calls are SQL statements issued by the database when executing user SQL statements. These calls degrade the database performance.

An indirect way to determine if there are data dictionary cache misses is to monitor the number of recursive calls with the **monitor** command.

1. At the SQLDBA prompt, enter:

```
monitor sessionstatistic
```

   The ORACLE Session Statistics Monitor screen displays.

2. Use the **TAB** key and press **F11** key to select "User."

   A display similar to the following appears:

```
                        ORACLE Session Statistics Monitor

          Starting Session ID:              Ending Session ID:

[ X ] User  [ ] Redo  [ ] Enqueue  [ ] Cache  [ ] ParServer  [ ] SQL

Statistic Name          Current     Average     Minimum     Maximum     Total
logons                        0        null        null        null         1
current logons                0        null        null        null         1
user commits               0.82        0.82        0.82        0.82        42
•
•
•
recursive calls            0.00        0.00        0.00        0.00         0
```

The "Total" column shows the cumulative value for each statistic since you began viewing the display. If the database does not continue to make recursive calls after startup, your data dictionary cache is probably large enough for your dictionary data. If the number of recursive calls accumulates while your application is running (and the value of recursive calls in the Total column increases), then there might be data dictionary cache misses.

You can check which parts of the data dictionary cache are filling up and are causing cache misses. Examine the cache activity by querying the V$ROWCACHE table. At the SQLDBA prompt, enter:

```
select parameter, count, usage, gets, getmisses from v$rowcache;
```

This command produces a display similar to the following:

```
PARAMETER            COUNT    USAGE    GETS    GETMISSES

                        50        0       0            0
                       300      265   38150         2764
                       200        0       0            0
•
•
•
```

The following table describes the columns of this query:

Table 2-1
V$ROWCACHE Query Table Columns

Column Heading	Description
PARAMETER	The name of the parameter that determines the number of entries in the data dictionary cache.
USAGE	Number of cache entries that contain valid data.
COUNT	Total number of entries in the cache.
GETS	Total number of requests for information on the data item.
GETMISSES	Number of data requests resulting in cache misses.

NOTE: For a detailed description of all columns presented in this and other dynamic performance tables, see the *ORACLE RDBMS Database Administrator's Guide Volume III Version 7.0.*

If any row in the table has a USAGE value that is significantly less than its COUNT value (USAGE is less than 80 percent of COUNT), you can reduce the number of cache entries to save memory. When USAGE exceeds 80 percent of COUNT, or if the ratio of GETMISSES to GETS for any dictionary cache continues to increase above 10 to 15 percent while the application is running, consider increasing the amount of memory available to the data dictionary cache.

To increase the available memory to the cache, increase the value of the initialization parameter **shared_pool_size**. For more information, see the *ORACLE RDBMS Performance Tuning Guide Version 7.0.*

Reduce Rollback Segment Contention

Rollback segments are allocations of space in the database that hold information needed to reverse, or undo, changes made by transactions. These segments are used for transaction rollback, read consistency, and recovery. Because ORACLE frequently accesses the database blocks that make up rollback segments, the segments can be subject to contention.

Examine contention for buffers in the buffer cache in the SGA and then determine if any buffer contention is due to rollback segment contention.

1. At the SQLDBA prompt, enter:

   ```
   monitor sessionstatistic
   ```

   The ORACLE Session Statistics Monitor screen displays.

2. Use the **TAB** key and press **F11** key to select "Cache."

   A display similar to the following appears:

```
                        ORACLE Session Statistics Monitor

          Starting Session ID:                Ending Session ID:

[ ] User  [ ] Redo  [ ] Enqueue  [X] Cache  [ ] ParServer  [ ] SQL

Statistic Name          Current    Average    Minimum    Maximum      Total

db block gets             38.46      38.36      38.46      38.46        500
consistent gets           95.31      95.31      95.31      95.31       1239
physical reads            41.15      41.15      41.15      41.15        535
buffer busy waits          0.00       0.00       0.00       0.00          0
```

To locate rollback segment contention, first examine contention for all buffers in the cache. The sum of the *consistent gets* and *db block gets* is the total number of requests for blocks. The *buffer busy waits* is the number of requests for buffers that result in waiting.

Obtain the ratio of buffer busy waits to the sum of *consistent gets* and *db block gets* in the statistics while your application is running. If the ratio is more than 10 or 15 percent, buffer contention might be affecting performance.

Determine what type of block is subject to contention by querying the V$WAITSTAT table. This table contains statistics that reflect block contention for different block classes. For example, you could enter the following:

```
select class, count, time from v$waitstat where class in
('system undo header', 'system undo block', 'undo header', 'undo block')
```

This command produces a display similar to the following:

```
CLASS                                COUNT          TIME
-------------------------------------------------------------------------------
system undo header
system undo block
undo header
undo block
```

The following table describes each block class:

Table 2-2
Block Classes in V$WAITSTAT Table

Block Class	Description
system undo header	The number of waits for buffers containing header blocks of the SYSTEM rollback segment.
system undo block	The number of waits for buffers containing blocks other than header blocks of the SYSTEM rollback segment.
undo header	The number of waits for buffers containing header blocks of non-SYSTEM rollback segments.
undo block	The number of waits for buffers containing blocks other than header blocks of non-SYSTEM rollback segments.

If most of the contention is for buffers containing either rollback segment header blocks or other rollback segments blocks, reduce contention by creating more rollback segments. See the *ORACLE RDBMS Performance Tuning Guide Version 7.0* for the recommended number of rollback segments.

Identifying Disk I/O Problems

After investigating the memory, the next step of the tuning methodology is to identify potential disk I/O bottlenecks.

Use the UNIX **sar** utility with the **-b** option to collect system activities. After you collect the data, analyze the **-b** option report to investigate the UNIX buffer activity. For detailed information on the **sar** report, see Chapter 4, "Performance Analysis Tools Reference."

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

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

If you use raw devices for the database files, the ORACLE disk I/O rates are in the "pread/s" and "pwrit/s" columns of the report. The majority of the disk I/O for the non-ORACLE processes is through the UNIX buffer cache. You can view this in the "bread/s," "lread/s," "%rcache," "bwrit/s," "lwrite/s," and "%wcache" columns. With this report, you can identify the total amount of disk I/O and how much ORACLE I/O (pread/s and pwrit/s) and non-ORACLE I/O (bread/s and bwrit/s) is through the raw devices.

NOTE: For more examples of using the **sar** command with other options, see Chapter 3, "Tuning the Disk Subsystem," in the COMPAQ TechNote *Performance Tuning for SCO UNIX on COMPAQ Systems.*

ORACLE disk I/O performance depends on the distribution of tablespaces across the logical disks on the COMPAQ Controller. The **sar** report gives only a system-wide I/O picture; it does not break down the I/O activity statistics to each logical disk. No performance tool with this capability is currently available.

Disk I/O Contention

Disk I/O contention is an important consideration in evaluating ORACLE RDBMS performance. Some important factors that contribute to disk I/O contention problems are as follows:

- I/O Distribution
- Buffer cache
- Data dictionary cache
- Database writer (DBWR)
- Sort area
- Data segment fragmentation

Tuning the I/O Distribution

If your system has more than two logical drives, I/O distribution is a significant performance consideration. The SMART Controller optimizes disk throughput with features such as bus mastering, data striping, parallel data transfers, simultaneous request servicing, optimized request management, and a four-megabyte cache. However, there can still be situations when the disk subsystem cannot keep up with the number of requests.

An I/O bottleneck occurs when requests from the device driver come to the SMART Controller faster than the controller can process them. Additional disk requests are placed ("queued") in the device driver, and are sent to the SMART Controller later. Meanwhile, outstanding requests sent from the device driver are also queued in the SMART Controller. When the SMART Controller becomes saturated with requests, disk performance degrades in proportion to the number of outstanding disk requests.

If your system performance is bound by the disk, use the **monitor** command to display I/O activity of all the available database files/devices. At the SQLDBA prompt, enter:

```
monitor file
```

The command produces a display with the following column headings:

	Request Rate	Batch Size	Response Time	Total Blocks
File Name	Read/s Write/s	blks/R blks/W	ms/Read ms/Write	Read Written

Look at the "Total Blocks" columns. If the sum of the "Read" and "Written" columns is evenly distributed for all the files/devices, the system has good I/O distribution. If one device shows a much higher number of I/Os than the other devices, you might need to relocate some of the files to a less active disk. For more information on distributing I/O, see Chapter 5, "Tuning I/O," in the *ORACLE RDBMS Performance Tuning Guide Version 7.0.*

Identify Hot Disks

You can also use this display to locate "hot" disks, which are disks with a high I/O rate. The definition of a high rate varies and depends on the system or hardware.

The total I/O rate for a single disk is the sum of "Request Rate Read/s" and "Request Rate Write/s" for all the ORACLE database files managed by the ORACLE instance on that disk. Determine this value for each of your disks and compare it to the I/O capacity of your disk.

Determine the total number of blocks written to each disk per second. For a single file, the number of blocks written per second is the product of Request Rate Write/s and Batch Size blks/W. The total number of blocks written to each disk is the sum of the blocks written to all of its files. You can then compare disks to determine which one has a high I/O activity.

Observe Disk Activities

Observe the disk activities by using the **sar** with the **-d** option. Disks holding database files and redo log files might also hold files that are not related to ORACLE.

The **sar-d** command produces a report with the following headings:

```
device    %busy    avque    r+w/s    blks/s    avwait    avserv
```

The "r+w/s" column shows the number of reads and writes per second per disk. Check the "%busy" column, also. A value of 60 - 70 or more indicates high disk activity. Try to reduce any heavy access to disks that contain ORACLE files. You can also check physical I/O for each file by querying the V$FILESTAT or X$KCFIO ORACLE tables.

Distribute hot files to less-active disk devices to balance disk I/O. You can move one entire hot file from a very active disk to a less active one.

Tuning the ORACLE Buffer Cache

The ORACLE buffer cache is in the SGA in shared memory. It temporarily holds these ORACLE data blocks:

- Tables
- Indexes
- Rollback segments
- Temporary segments

The ORACLE buffer cache has a great effect on disk I/O performance. Because the cache is in shared memory, all ORACLE processes can access it once a process reads a data block from the database file. If the ORACLE buffer cache is not adequately configured, the unnecessary disk I/O degrades system performance.

Identifying an ORACLE Buffer Cache Problem

Use the **monitor** command to watch the buffer cache activity of the ORACLE processes. At the SQLDBA prompt, enter:

```
monitor systemio
```

This command produces a display similar to the following:

ORACLE System I/O Monitor

		----------Interval----------						----------Cumulative----------					
Process ID	Session ID	%logical reads		%physical reads		%logical writes		%logical reads		%physical reads		%logical writes	
		0	100	0	100	0	100	0	100	0	100	0	100
Totals		98		98		1348		7782		363		4665	
										Hit Ratio		0.95	

The "Process ID" here is an ORACLE Process ID, not a UNIX Process ID. If you have more processes than can be displayed on one screen, they display sequentially at specified intervals. The important statistic is the "Hit Ratio" in the "Cumulative" column. The Hit Ratio is defined as:

```
Hit Ratio = (logical reads - physical reads) / logical reads
```

The Hit Ratio is an important indicator of the efficiency of buffer cache usage. If the ratio is above 0.9 (90 percent cache hit) you configured enough buffers in the cache. If the ratio is below 60 or 70 percent, increase the number of buffers in the cache by increasing the parameter **db_block_buffers**.

Determining the Number of Cache Buffers to Add

To help you determine how many block buffers to add for your application, you can query the X$KCBRBH table. This table contains the cache-block buffer statistics.

Set the **db_block_lru_extended_statistics** parameter to TRUE. Then you can query the X$KCBRBH table. The result of the query helps you determine the effect on the hit ratio when you add or reduce the buffers by a certain number. For more information on the X$DCBRBH table, see Chapter 4, "Tuning Memory Allocation," of the *ORACLE RDBMS Performance Tuning Guide Version 7.0*.

Tuning the Data Dictionary Cache

The data dictionary cache (also in the SGA) holds reference information about database objects such as rights and privileges of ORACLE users. ORACLE frequently accesses the data dictionary during the parsing of SQL statements. If the information is not in the cache, ORACLE must generate SQL statements to query the data dictionary tables in the database, causing recursive calls. Recursive calls degrade ORACLE performance.

Identifying a Data Dictionary Cache Problem

Use the **monitor** command to see the number of recursive calls.

1. At the SQLDBA prompt, enter:

   ```
   monitor sessionstatistic
   ```

   The ORACLE Session Statistics Monitor screen displays.

2. Use the **TAB** key and press **F11** key to select "User."

   A display similar to the following appears:

```
                    ORACLE Session Statistics Monitor

          Starting Session ID:              Ending Session ID:

[X] User  [ ] Redo  [ ] Enqueue  [ ] Cache  [ ] ParServer  [ ] SQL

Statistic Name                  Current   Average   Minimum   Maximum     Total

cpu used by this session                                                        0
cumulative logons                                                              12
cumulative opened cursors                                                     538
current logons                                                                  6
current open cursors                                                            1
max session memory                                                         446660
recursive calls                                                              8557
recursive cpu usage                                                             0
session connect time                                                            0
```

A steady increase in the recursive calls statistics might imply that some of the data dictionary caches are too small. To verify this, query the dynamic performance table V$ROWCACHE.

Determining Which Data Dictionary Cache to Add

If some of the data dictionary caches are too small, the next step is to determine which ones. Then you can adjust the **dc_xxxx** parameters in *init.ora* to increase the cache hit.

The system table V$ROWCACHE holds statistics on data dictionary activity. It has one row for each dictionary cache. To query the table, enter the following command:

```
select parameter, gets, getmisses, count, usage from v$rowcache
```

This command produces a display similar to the following:

PARAMETER	COUNT	USAGE	GETS	GETMISSES
	50	0	0	0
	300	265	38150	2764
	200	0	0	0
•				
•				
•				
dc_users	50	3	914	3

For dictionary caches that exhibit continued "GETMISSES," increase the values of the appropriate parameters in the *init.ora* file. Note that when the database is starting up, misses should occur, and getmisses should stop increasing after startup.

Use the "USAGE" and "COUNT" columns to decide whether to reduce or increase the appropriate parameters. If the usage figure is significantly below the value of count, reduce the cache size. The "PARAMETER" column shows which parameter to adjust. If usage is close to the count value, try increasing the cache size.

Tuning the Database Writer

The database writer (DBWR) writes data blocks from the buffer cache in SGA to disk. There are two lists of buffers in the cache, the "dirty list" and the least-recently-used (LRU) list. The dirty list holds modified buffers that have not been written to disk. The LRU list holds free buffers, in-use ("pinned") buffers, and dirty buffers that have not yet been moved to the dirty list.

DBWR starts writing dirty buffers to disk under these circumstances:

- When a user process finds that half of the number of **db_block_write_batch** buffers are on the dirty list
- When a user process scans **db_block_max_scan_cnt** buffers and cannot find a free one
- When a timeout occurs at about every three seconds
- When a checkpoint occurs

We recommend that you use asynchronous I/O to obtain maximum performance of ORACLE. With asynchronous I/O, the **dbwr** process sends multiple writes to multiple drives without waiting for the previous writes to finish. It can use all disks simultaneously. With 10 disks, for example, 10 asynchronous writes can take place in the same amount of time as a single synchronous write.

To turn on asynchronous I/O, set **db_writers=1** in the *init.ora* file. For more information about this topic, see the Compaq TechNote, *The Compaq, SCO, and ORACLE7 Database Server*.

Sort Area

The sort area is the amount of memory allocated to a user process for sort/merge operations such as **create index**, **select distinct**, **order by**, **group by**, and certain **join** operations. If the amount of data to be sorted does not fit in the sort area, the data are divided into smaller pieces. Each piece is sorted separately. After each intermediate sort, the result is stored in the temporary segment on disk. This increases disk I/O and might cause a performance problem.

Use the **monitor** command to look at the database file I/O statistics. At the SQLDBA prompt, enter:

```
monitor file
```

This command displays all the database files accessed by the ORACLE RDBMS, along with the statistics reflecting their I/O activity. The display contains the following column headings:

	Request Rate		Batch Size		Response Time		Total Blocks	
File Name	Read/s	Write/s	blks/R	blks/W	ms/Read	ms/Write	Read	Written

The column, "Request Rate Read/s" is the average number of reads from each database file, per second. The column, "Request Rate Write/s" is the average number of writes to each database file per second. "Batch Size blks/W" indicates the average number of data blocks written to each database file in a single write.

The total I/O rate for a single disk is the sum of Request Rate Read/s and Request Rate Write/s for all ORACLE database files managed by the ORACLE instance on that disk. Determine this value for each of your disks and compare it to the I/O capacity of your disk.

Also determine the total number of blocks written to each disk per second. For a single file, the number of blocks written per second is the product of Request Rate Write/s and Batch Size blks/W. The total number of blocks written to each disk is the sum of the blocks written to all of its files. You can then compare disks to determine which disk has a high I/O activity.

This option gives the I/O rate per disk file (raw device). The important statistic here is the amount of disk I/O attributed to the disk file (raw device) that contains the tablespace for the temporary segment. Monitor this when the system is doing heavy sorting. For best performance, I/O should be generally distributed among the database devices.

If excessive I/O comes from accessing the temporary segment, the system is performing too many intermediate sort runs. If your system is experiencing excessive I/O, try increasing the **sort_area_size** parameter.

Data Segments Fragmentation

Once the database is in operation, space is dynamically allocated and deallocated by creating, dropping, and modifying database objects. The database begins to fragment. Fragmentation also degrades disk performance.

Generally, you cannot totally avoid fragmentation. You can control it by carefully planning the storage requirements when you create a data table. The following table lists the storage parameters:

Table 2-3
Data Table Storage Parameters

Parameter	Description
initial	The size, in bytes, of the first extent of the segment. The default value is 10,240 bytes. ORACLE rounds the value to the number of ORACLE blocks.
next	The size of the second and subsequent extents. The default value is 10,240 bytes.
pctincrease	The percentage of increase for the next extent. This parameter takes effect beginning with the third extent. A larger value ensures more contiguous storage space for the expanded extents. The default is 50.
minextents	The minimum number of extents when the segment is created. Together with **initial**, this parameter determines the starting storage allocation. The default is 1.
maxextents	The maximum number of extents for this segment. The default size varies, depending on the data block size.

Adjust the **initial**, **next**, **minextents**, **maxextents**, and **pctincrease** storage parameters. Use these in conjunction with **pctfree** and **pctused** to control fragmentation.

Fragmentation can increase the number of recursive calls. To monitor the recursive calls, use the **monitor** command. Select a PID of 0 to display the system data. At the SQLDBA prompt, enter:

```
monitor user 0
```

Because other factors might also cause recursive calls, query some system and dynamic performance tables to verify that there is a segmentation problem. Query the **sys.dba_segments** table and look for segments with more than 3 extents. A segment with too many extents might have a fragmentation problem. Query the **dba_space_free** and **dba_space_summary** tables to obtain a report on the free space of each tablespace.

To eliminate fragmentation, Oracle recommends that you perform a full database export, drop the database, recreate it, and then import the tables. If you determine that you have only a small number of fragmented tables, you can export the tables, drop the tables, recreate them, and import the table data. By selecting compression, you can compress the table and its indexes into one extent for each structure.

Chapter 3
Tuning the System Processor and Reducing Oracle Contention

The final step of the tuning methodology is to determine if your system is system processor-bound. A processor-bound system occurs when there is more database activity than the system processor can handle quickly and when the database is not tuned to avoid unnecessary contention for ORACLE resources.

Identifying System Processor Problems

If processes are contending for ORACLE resources, additional system processor power should decrease the contention time. However, first determine that the system is not I/O bound. Additional system processor power will not improve the performance of an I/O-bound system.

UNIX sar Command

Use the UNIX **sar -u** command to identify potential system processor bottleneck problems. The **sar -u** command produces a report that gives only the percentage of system processor use and a system-wide system processor loading picture. Use the UNIX **ps** command to identify the number and system processor usage of each active ORACLE and non-ORACLE process so you can identify if the system processor problem is caused by ORACLE processes.

The **sar -u** command produces a report with the following headings:

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

As a general rule, your applications should spend more time on user time ("%usr") than on system time ("%sys"). The "%wio" and "%idle" columns indicate the percentage of time the system processor is waiting for I/O, and is idle.

Your target percentages should be:

- %usr - 60 percent
- %sys - 20 percent
- %wio - 10 percent
- %idle - 10 percent

Some causes of large system times are too much context switching, too many interrupts, or too many processes running. You might be running several programs that use the system inefficiently. If the %wio column indicates a value that is a significant portion of system processor activity, investigate your I/O performance. If your system is loaded heavily and %idle is high, you might have memory or disk problems.

NOTE: For more information on the **sar** command, see Chapter 4, "Performance Analysis Tools Reference."

If you have a multiprocessor environment, you can improve system performance by distributing the workload evenly across the system processors.

UNIX mpstat Command

If you have a multiprocessor environment, you can use the UNIX **mpstat** command to monitor the processor loads. The **mpstat** command produces a display with the following headings:

```
system    user status    sys    cs    int    tr
```

The following table lists the values for these headings:

Table 3-1
UNIX mpstat Column Headings

Column Heading	Description
system	The system processor number (CPU1, CPU2, and so on).
user	The kernel/user code.
status	The status of the system processor. Possible states: ACTIVE Processor available to run any process. STATIC Processor can only run processes specifically designated for that processor. INACTIVE Processor runs no processes.
sys	The number of system calls per period of time.
cs	The number of context switches (the number of changes of running processes).
int	The number of interrupts.
tr	The number of traps.

For more information on the **mpstat** command, see Chapter 4, "Performance Analysis Tools Reference."

Reducing ORACLE Contention

If your system is system processor-bound, first try to reduce ORACLE contention. After this, you can decide whether to reduce the number of ORACLE users, reschedule their access, or upgrade the system processor. This section provides methods to detect ORACLE contention and discusses steps you can take to reduce it.

Rollback Segment Contention

Rollback segments are allocations of space in the database that hold information necessary to reverse or undo changes made by transactions. Rollback segments are used for transaction rollback, read consistency, and instance recovery. Because ORACLE frequently accesses the database blocks that make up rollback segments, rollback segments can be subject to contention.

To detect rollback segment contention, use the **monitor** command. At the SQLDBA prompt, enter:

```
monitor statistics cache
```

The command displays cache statistics similar to the following:

Statistic Name	CUR	AVG	MAX	MIN	TOT
db block gets	38.46	38.36	38.46	38.46	500
consistent gets	95.31	95.31	95.31	95.31	1239
physical reads	41.15	41.15	41.15	41.15	535
•					
•					
•					
buffer busy waits	0.00	0.00	0.00	0.00	0
•					
•					
•					

The relevant statistics are "db block gets," "consistent gets," and "buffer busy waits." The sum of the first two statistics represents the total number of block gets, including those found and not found in the SGA. The buffer busy waits statistic is the number of requests that result in waiting.

$$\text{Contention ratio} = \frac{\text{buffer busy waits}}{\text{db block gets} + \text{consistent gets}} * 100\%$$

If the contention ratio is greater than 10 or 15 percent, then you might have a buffer contention problem. Determine what type of block is subject to contention by querying the V$WAITSTAT table. See the next topic, "Query the Dynamic Performance Table."

Another important cache statistic is "physical reads." The sum of db block gets and consistent gets represents reading within SGA. The sum of the three statistics represents the total number of blocks read.

$$\text{Hit ratio} = \frac{\text{db block gets} + \text{consistent gets}}{\text{db block gets} + \text{consistent gets} + \text{physical reads}} * 100\%$$

The Hit Ratio should be always greater than 90 percent. If not, increase the number of data buffers.

Query the Dynamic Performance Table

The dynamic performance table V$WAITSTAT collects block contention statistics. For detailed descriptions of the columns of this table, see Appendix F, "Data Dictionary Reference," of the *ORACLE RDBMS Database Administrator's Guide Volume III Version 7.0*.

To determine the total number of waits for each class of block, enter the following query:

```
select class, sum(count) total_waits
from sys.v$waitstat
where operation = 'buffer busy waits'
and class in ('data block','undo segment header','undo block')
group by class
```

Waits for undo block or undo segment header blocks mean there is rollback segment contention.

Reducing Rollback Segment Contention

You can reduce rollback segment contention by creating additional rollback segments until you reduce the number of waits from the query list. To cache rollback segments, make them small. However, if most of the transactions change large amounts of data or if you have long running queries that need rollback segments for read consistency, you might want to create larger rollback segments.

For details on adding rollback segments, refer to Chapter 5, "Data Blocks, Extents, and Segments," and Chapter 6, "Tablespaces and Data Files," of the *ORACLE RDBMS Database Administrator's Guide Volume III Version 7.0*.

Free List Contention

Each table has one or more free lists. Free list contention can occur if many ORACLE processes are trying to insert rows into the same table simultaneously. Each process must access a free list. If there are fewer free lists than processes, some of the processes might have to wait for a free list.

First, identify buffer contention with the **monitor statistics cache** command, and look at the ratio of *consistent gets* and *db block gets* to *buffer busy wait*. If this ratio is over 10 percent, query the V$WAITSTAT table with the following commands:

```
select class, sum(count) total_waits
from sys.v$waitstat
where operation = 'buffer busy waits'
and class in ('data block','undo segment header','undo block')
group by class
```

If the number of data block waits is proportionally large compared to that of undo block waits, there might be free list contention.

You can add more free lists to reduce free list contention. Refer to Chapter 7, "Additional Tuning Considerations," of the *ORACLE RDBMS Performance Tuning Guide Version 7.0*.

Chapter 4
Performance Analysis Tools Reference

This chapter contains information on the UNIX **sar** and **mpstat** utilities and the ORACLE **monitor** utility. For a detailed review of these utilities, see the appropriate UNIX and ORACLE documentation.

The SAR Utility

Purpose: Reports system activity

Syntax: sar[A-aBbcdghmnpqRruvwy] [-A] [-o *file*] *t* [*n*]

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

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

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

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

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

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

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

Table 4-1
sar Command Options and Report Information

Option	Description	Report Column Headings
-A	A summary of all reports.	
-a	Reports use of file access system routines.	*iget/s* Number of files located by i-node entry per second. *namei/s* Number of filesystem path searches per second. *dirblk/s* Number of directory block reads issued per second.
-B	Reports additional buffer cache activity.	*cpybuf/s* Number of copy buffers required per second. *slpcpybuf/s* Number of times necessary to "sleep" while waiting for a copy buffer.

Continued

Table 4-1 *Continued*

Option	Description	Report Column Headings
-b	Reports buffer cache activity.	*bread/s*, *bwrit/s* Average number of block reads or writes, per second, between system buffers and disk or other block devices. *lread/s*, *lwrit/s* Average number of logical block reads and writes, per second, from system buffers. *%rcache*, *%wcache* Percentage of logical reads and writes found in buffer cache. *pread/s*, *pwrit/s* Average number of physical reads and writes (block reads and writes), per second.
-c	Reports system calls.	*scall/s* System calls of all types, per second. *sread/s, swrit/s, fork/s, exec/s* Specific system calls, per second. *rchar/s, wchar/s* Characters transferred by read and write system calls, per second.

Continued

Table 4-1 *Continued*

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

Continued

Table 4-1 *Continued*

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

Continued

Table 4-1 *Continued*

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

Continued

Table 4-1 *Continued*

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

Continued

Table 4-1 *Continued*

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

The MPSTAT UTILITY

The SCO UNIX MPX software includes a processor load status display utility, **mpstat**(MP). This utility displays processor activity information on your screen for each of the processors installed on your system. It allows you to verify that the system load is balanced across all available processors.

Every second, **mpstat** takes a "snapshot" of system resource use, and displays the percentage of system (kernel) code and user code running on each processor, plus processor idle time. The **mpstat** utility also displays the processor status, the number of system calls, context switches, interrupts, and traps that have occurred on each processor in the past one-second interval.

Syntax: mpstat[-bcCPUhoxV]

The **mpstat** utility is highly processor resource intensive. Allowing **mpstat** to free run on any processor has a noticeable effect on the processor load results that you obtain. We recommend that you invoke **mpstat** using the **-c** and **-h** options. Alternatively, lock **mpstat** onto a specified processor and hide the statistic of that processor. However, this procedure means that the system is effectively running with one less system processor.

The following table describes the options available for the **mpstat** utility.

Table 4-2
mpstat Options Summary

Option	Description
-b	Use black and white for screen colors. This option is useful when running under X Windows.
-c *CPU*	Locks **mpstat** program on processor *CPU*, where *CPU* is replaced with the number of the processor. Cannot be used with the **-x** option.
-h	Hides the statistics of the locked processor.
-o	Displays the Options window.
-x	Allows **mpstat** to free run on any processor. Default option. Cannot be used with the **-c** option.
-V	Displays **mpstat** version number. If used with any other options, the other options are ignored when the command is executed.

The MONITOR UTILITY

When you use the **monitor** utility, you produce a continuously updated display of database activity. You view this information on a specific type of "monitor" that you access with the appropriate option.

Syntax: monitor[*options*]

The following table describes the options available for the **monitor** utility.

Table 4-3
Monitor Options Summary

Option	Description
Cycle	Sets the data display screen refresh cycle to *n* seconds, with *n* ranging from 1 to 3600. We recommend a 5-second minimum because of system processor overhead. The refresh cycle time also applies to data that has more than one screen page.
File I/O	Shows the amount of disk I/O activity for each database file or raw device. This is useful in checking the disk I/O distribution. Knowing on which logical drive these files are located can help you determine if you want to move the tablespaces to tune I/O distribution.
I/O	Shows the ORACLE processes and their I/O characteristics. **monitor** uses the cumulative logical and physical reads to calculate the hit ratio of the ORACLE data buffer cache. **monitor** displays the ratio on the bottom line of the screen.

Continued

Table 4-3 *Continued*

Option	Description
Latch	Displays the latch statistics. Latches are short-lived locks used to protect the database's internal resources. The important figures are Timeouts and Total; the ratio should not be more than 10 percent. Latches do not generally require tuning.
Lock	Displays the locks held by processes on user or data dictionary tables. You can use this information to observe how processes lock the database resources.
Process	Displays the processes that are connected to ORACLE. Use the ORACLE PIDs to identify the ORACLE processes in other Monitor options. With the system PID of the ORACLE processes, you can obtain operating system level information using the UNIX **ps -ef** command.
Rollback	Displays the current status of all the rollback segments. The header column figure is important, because the user process needs to access the rollback segment header to find the address of the rollback data in the rollback segment. The ratio of header waits per second (Waits/sec) to gets per second (Gets/sec) should be less than 1 percent. If the extents value is high, add new segments or increase the size of current segments.
Spool	Specifies an output file to obtain a copy of the **monitor** output.

Continued

Table 4-3 *Continued*

Option	Description
Statistics	Displays information on different classes of statistics. Due to the number of statistics, you should also specify the class of data which interests you. The classes are: user Displays data on all the user processes. Recursive calls and current open cursors are the important statistics. enqueue Displays information on processes that attempt to obtain locks on database objects. Enqueue deadlocks is an important figure; a non-zero value indicates that a deadlock has occurred. cache Displays information on the data cache in the ORACLE System Global Area (SGA). It provides important data for tuning the data buffer cache and data dictionary cache. redo Provides statistical data on how read information is handled from its creation in the user buffer area to the point at which it is written to the redo log file. Redo operation does not generally require tuning.

Continued

Table 4-3 *Continued*

Option	Description
Table	Displays information on processes and the tables they are accessing. Obj# is an important figure; it gives the object number of the table being used.
User	Displays information on user processes. Data similar to those from the process option display. The information is *not* intended to be used as a tuning aid.

Index

SPECIAL CHARACTERS

$

C

D

F

G

H

I

L

M

N

O

P

R

S

T

U

V

X