Embarcadero Performance Center 2.7: Oracle Expert Guide

Embarcadero Technologies, Inc.
100 California Street, 12th Floor
San Francisco, CA 94111 U.S.A.

Contents

Oracle Expert Guide

This section includes expert help for all Oracle categories and statistics in Performance Center views. For detailed information on using the application, see Using Performance Center. This guide includes the following sections:

- Home View Statistics
- Contention Page Statistics
- I/O Page Statistics
- Memory Page Statistics
- Network Statistics
- Objects Page Statistics
- OS Page Statistics
- Session Details
- Space Page Statistics
- Users Page Statistics
- Other Statistics

Home Statistics

The Home view lets you review availability and overall performance of all monitored databases from a single window. The Home page includes the following sections:

- Contention Vital Signs
- I/O Vital Signs
- Memory Vital Signs
- Network Vital Signs
- Space Vital Signs
- Users Vital Signs

Related Topics

I/O Page Statistics

Memory Page Statistics

Objects Page Statistics

OS Page Statistics

Space Page Statistics

Top SQL Page Statistics

Users Page Statistics

Network Statistics

Contention Statistics

Other Statistics

Memory Vital Signs

The following memory statistics are on the Oracle Home view:

- Buffer Cache
- Dictionary Cache
- Library Cache
- Memory Sort Ratio

Contention Vital Signs

The following contention statistics are on the Oracle Home view:

- Enqueue Waits
- Redo Log Space Wait Time
- Users Blocked
- Users Waiting

Users Waiting

- Metrics
- Troubleshooting

User connections that are waiting on a system generally occur for two reasons:

1. A process waits because a requested resource is not available.
2. A process waits for Oracle to perform a prerequisite task for its given operation.

These two wait causes are worth your time and investigation; idle waits (processes waiting because they have no work) should not worry you at all.

Metrics

To determine the actual wait causes user connections are experiencing, drill down from the global count of users waiting into the actual system and user wait details of your target database. This lets you locate the exact causes of currently experienced waits.

Troubleshooting

If you find a problem, drill down into wait details to determine whether the waits are resource related.

Users Blocked

On a small system, a single blocking user has the potential to stop work for nearly all other processes. On larger systems, this can cause major headaches. Although Oracle supports unlimited row-level locking, blocking lock situations do occur. User processes holding exclusive locks and not releasing them via a proper COMMIT frequency, generally cause most blocks.

Metrics

You should investigate any blocking lock statistic indicator above zero to prevent a mushrooming situation.

Troubleshooting

You can quickly remedy a blocking lock situation by issuing a KILL against the offending process. This eliminates the user's stranglehold on the accessed objects and lets other user processes complete. Tools like Performance Center make it easier to discover the blocked lock situation, the tricky part is preventing the blocking lock situation in the first place.

The culprit of blocking lock scenarios is usually the application design, or the SQL used within the application itself. Properly coding an application to reference database objects in an efficient order and then using the right SQL to get the job done, is an art. Most DBAs who have had to face Oracle Forms applications have suffered through the dreaded SELECT…FOR UPDATE statements that place unnecessary restrictive locks on nearly every read operation, know all too well that good coding practice is important. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible, something not always easy to do.

Enqueue Waits

An enqueue is an advanced locking device that lets multiple database processes share certain resources. Enqueue waits typically occur when sessions wait to be granted a requested lock. Sometimes these locks are internal Oracle locks while other times they could be locks for rows of data in a table.

> **NOTE:** Enqueues are issued implicitly by Oracle.

Metrics

You should investigate any enqueue waits that read consistently above one or more (delta statistics).

Troubleshooting

Removing contention for enqueues is usually an application design issue. Many enqueue waits are either contention for certain rows in the database or the result of database-initiated lock escalation. You should examine the use of indexes to make sure all referencing foreign keys are indexes and that your SQL does not tarry over rows in the database during modification operations. If it does, you should tune your SQL.

Enqueue waits can also be the result of space management tasks (such as objects extending) and disk sorts (mainly in tablespaces that do not make use of the TEMPORARY tablespace parameter).

Redo Log Space Wait Time

- Troubleshooting

The Oracle RDBMS is able to manage recovery by recording all changes made to a database through the use of redo log files. Oracle writes modifications made to a database to the redo log files, which you can archive off to another medium for disaster recovery. The background process that performs these operations is Oracle's Log Writer (LGWR). There is a buffer area in Oracle's System Global Area (SGA) that is used to reduce redo log file I/O, whose size, or lack thereof, can affect performance in a busy system. Sometimes a user process must wait for space in the redo log buffer. Oracle uses the log buffer to cache redo entries prior to writing them to disk. If the buffer area is not large enough for the redo entry load, waits can occur. While waits can be an important indicator of contention, a better measure is the amount of wait time that your users experience.

Metrics

The two main numbers to watch are:

1. Redo log space requests
2. Redo log wait time

If either statistic strays too far from zero, you may want to increase the log_buffer parameter and add more memory to the redo log buffer.

Troubleshooting

If you find a problem, do the following:

1. Edit the Init.ora file for the database.
2. Increase the amount of log_buffer to a higher value (take care not to over-allocate; ensure enough free memory exists on server before increasing value.
3. Cycle the Oracle server when possible to allow the new value to take effect.
4. Monitor new value to see if performance improves.

When adjusting the log_buffer, make sure that the amount is a multiple of the block size. Otherwise, on database startup, Oracle returns an error stating that you have entered an invalid amount for the redo log buffer.

> **NOTE:** If you make the log_buffer parameter smaller than its default size for a given platform, Oracle will silently increase it.

I/O Vital Signs

The following I/O statistics are on the Oracle Home view:

- Logical Changes
- Logical Reads
- Physical Reads
- Physical Writes

Physical Reads

This value reflects total number of physical reads performed on all datafiles since the last refresh.

Metrics

Large numbers of physical reads may indicate that your buffer cache is too small. You should examine the buffer cache hit ratio to determine the overall effectiveness of logical vs. physical I/O.

Physical Writes

This value reflects total number of times the DBWR process has performed writes to various database datafiles since the last refresh.

Metrics

Wait events related to I/O activity are good indicators of physical I/O problems. These events include db file parallel write and db file single write. You can view both of these events in Performance Center.

Logical Reads

This value reflects total number of db block gets and consistent gets (data read from memory) since the last refresh.

Metrics

Large numbers of logical reads may not indicate a problem, although extremely large amounts could indicate a runaway process. You should examine the buffer cache hit ratio to determine the overall effectiveness of logical vs. physical I/O. You could also examine the User I/O Detail view to see if any one process is churning up a lot of logical I/O.

Logical Changes

This is the total number of changes that were made to all blocks in the SGA that were part of an update or delete operation. These changes generate redo log entries and are permanent if the transaction is committed. The number of logical changes is an approximate indication of total database work.

Metrics

None.

Users Vital Signs

The following users statistics are on the Oracle Home view:

- Active Connections
- Current Locks
- Open Cursors
- Total Connections

Total Connections

- Metrics
- Troubleshooting

This statistic represents the total number of open threads, or connections, currently reported in the database. This number includes both active and inactive processes.

Metrics

You should view the total number of sessions in light of the maximum number of processes allowed to connect to Oracle. The processes parameter in the Init.ora file specifies the maximum number of operating system user processes that can simultaneously connect to an Oracle Server. Checking the Users performance category view can show just how close the system is coming to running out of processes.

Troubleshooting

If the total number of connections approaches the processes limit, do the following:

1. Edit the Init.ora file for the database.
2. Increase the amount of processes to a higher value.
3. Cycle the Oracle server when possible to allow the new value to take effect.

Active Connections

This statistic represents the total number of active and open threads, or connections, reported in the database. The number displays the number of processes actively performing work.

Metrics

None.

Current Locks

This statistic displays the total number of locks obtained by processes in the database.

Metrics

With respect to locks, the main thing to watch is that all DML locks currently held on the system do not approach the dml_locks limit specified in the Init.ora file. The parameter, dml_locks, limits how many locks can exist on the system at one time.

Troubleshooting

If the total number of locks approaches the dml_locks limit, do the following:

1. Ensure that user processes are efficiently using locks and are committing frequently to avoid excessive lock hold times before editing the Init.ora file.
2. Edit the Init.ora file for the database.
3. Increase the amount of dml_locks to a higher value.
4. Cycle the Oracle server when possible to allow the new value to take effect.

Space Vital Signs

The following space statistics are on the Oracle Home view:

- Free Space
- Used Space
- Tablespace Low on Space

Used Space and Free Space

- Metrics
- Troubleshooting

These statistics represent the total used and free space available in all tablespaces/datafiles in the database. Although this is good to know, you need a more detailed listing by tablespace to determine where any actual space shortages exist in the database.

You can view this information in the Performance Center Space performance category view.

Metrics

If any one tablespace begins to approach 90% used, the DBA should take action to prevent any future space allocation errors.

Troubleshooting

There are a couple of things a DBA can do to prevent a tablespace from running out of available free space:

1. Turn AUTOEXTEND on for the underlying tablespace's datafiles. This allows them to automatically grow when free space in the datafile has been exhausted.
2. Using the ALTER TABLESPACE…ADD DATAFILE... command, you can manually add a new datafile to a tablespace that is about to run out of available free space.

Tablespace Low on Space

- Metrics
- Troubleshooting

This statistic gives you a count of tablespaces with less than 10% free space.

Metrics

If any tablespace free space percent amount goes below 10%, you should take action to ensure that the tablespace does not run out of available free space.

NOTE: The default notification for Tablespace Low on Space is the designated Emergency Contact. If you do not want the notification to go to the designated Emergency Contact, you should create a Tablespace Low on Space notification.

Troubleshooting

There are two things you can do to ensure that a tablespace does not run out of available free space:

1 First, you should look into the use of Oracle's AUTOEXTEND feature. AUTOEXTEND lets you give an Oracle tablespace the ability to auto-grow when it has exhausted the free space contained within. You can let a tablespace grow in an unlimited fashion or put constraints on it to stop at a certain point. You can also dictate how much more free space the tablespace gets each time it needs more space than is available. However, AUTOEXTEND enabled for a tablespace does not mean that you cannot run out of space. Remember you still have the physical server limitations to contend with. Make sure you (or your sysadmin) keep a careful eye on the server drives that house your Oracle database files for available free space.

2 If the free space on a server drive nears its limit, disable AUTOEXTEND for the datafile(s) that are on that drive, and use the old-fashioned ALTER TABLESPACE... ADD DATAFILE command to place a new datafile for the tablespace on another drive that has more free space to offer.

TIP: AUTOEXTEND is not a replacement for proactive space planning on the part of the DBA. When the database needs extra space and AUTOEXTEND is activated by Oracle, you take a performance as Oracle allocates more space for the tablespace. Avoiding this type of extension aids performance, albeit in a small way.

Network Vital Signs

The following network statistics are on the Oracle Home view:

- Bytes from Client
- Bytes from DBLink
- Bytes to Client
- Bytes to DBLink

Bytes to Client

This statistic is the total number of bytes sent over the network to all Oracle client machines from the database since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Out of the ordinary numbers should raise alarms as possible network saturation could occur.

Bytes from Client

This statistic is the total number of bytes sent over the network to the Oracle database from all client machines since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Numbers out of the ordinary should raise alarms as possible network saturation could occur.

Bytes to DBLink

This statistic is the total number of bytes sent by all client machines over the network to any database link since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Numbers out of the ordinary should raise alarms as possible network saturation could occur.

NOTE: Database link traffic could experience longer response times depending on the distributed nature and setup of the involved databases.

Bytes from DBLink

This statistic is the total number of bytes received by all client machines over the network from any database link since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Numbers out of the ordinary should raise alarms as possible network saturation could occur.

NOTE: Database link traffic could experience longer response times depending on the distributed nature and setup of the involved databases.

- Metrics
- Troubleshooting

The buffer cache hit ratio is an indicator of how often user requests for data are satisfied through memory vs. being physically read from disk. Data read from memory produces user response times many times faster than when that same data is read from disk. Keeping physical I/Os to an absolute minimum is one of the Oracle buffer cache's purposes in life.

The table below describes the key counters Performance Analyst uses to calculate the buffer cache hit ratio:

Key Counter	Description
DB BLOCK GETS	Data read from memory for DML operations.
CONSISTENT GETS	Data read from rollback segments in memory.
PHYSICAL READS	Data read physically from disk.
Direct Reads	Data read physically from disk that bypasses the buffer cache. Direct reads are filtered out of overall physical reads so an accurate cache hit ratio can be determined.

Dividing the data read from memory by data read from disk yields the cache hit ratio.

NOTE: This statistic is also available on the Memory home page.

Metrics

To help ensure excellent performance, you want to keep your cache hit ratio in the neighborhood of 90% or higher. However, every database has its own ‘personality’ and can exhibit excellent performance with below average readings for the cache hit ratio. Excessive logical I/O activity can produce a very high cache hit ratio while actually degrading overall database performance.

Investigate consistent low readings of 60% or less.

NOTE: For Oracle8i or earlier, the adjustment of the db_block_buffers tuning parameter is required. For Oracle9i and later, the db_cache_size parameter is the parameter that needs attention. Any increases in db_block_buffers to take effect on Oracle8i or earlier, the database must be cycled. The db_cache_size parameter in Oracle9i or later, however, is dynamic and can be altered without stopping and starting the database instance.

Troubleshooting

If a problem is found in Oracle8i or earlier, do the following:

- Edit the Init.ora file for the database.
- Increase the amount of db_block_buffers to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor the new value to see if performance improves.

If a problem is found in Oracle9i or later, do the following:

- Increase the size of the db_cache_size parameter through use of the ALTER SYSTEM SET db_cache_size command value (take caution to not over-allocate; ensure enough free memory exists on server before increasing value).
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

Library Cache Hit Ratio

Oracle's shared pool offers a number of tuning possibilities and is made up of two main memory areas:

- Library Cache
- Data Dictionary Cache

The library cache hit ratio offers a key indicator in determining the performance of the shared pool. It holds commonly used SQL statements - basically database code objects. The library cache hit ratio shows how often SQL code is reused by other database users vs. the number of times an SQL statement is broken down, parsed, and then loaded (or reloaded) into the shared pool.

You can improve performance by encouraging the reuse of SQL statements so expensive parse operations can be avoided. The library cache assists this tuning effort.

NOTE: This statistic is available on the Home page, the Memory home page and on the Library Cache tab of the Memory Detail.

Metrics

A high library cache hit ratio is a desirable thing. Strive for a hit ratio between 95-100%, with 99% being a good performance benchmark for code reuse.

> **NOTE:** When a database is first started, the library cache hit ratio is not at an optimal level because all code being used is relatively new, and as such, must be parsed and placed into the shared pool. If, however, after a solid hour or two of steady database time, the library cache hit ratio has not increased to desirable levels, increase the shared_pool_size parameter.

Other red flags that can indicate a too small shared pool include:

- A wait count for the event 'latch free' of 10 or greater.
- The library cache wait count of two or greater.

These indicators can be tracked with Performance Analyst's Bottleneck and Wait detail views.

You can improve the library cache hit ratio by encouraging SQL code reuse through the implementation of bind variables. Discouraging hard coded literals in application code and instead making use of variables bound at run time aids in the reuse of SQL code that is maintained in Oracle's shared pool.

> **NOTE:** Bind variables can have an affect on the cost-based optimizer though.

A second way is to pin frequently used code objects in memory so they are available when needed, by using the system supplied DBMS_SHARED_POOL package. You can use Performance Analyst to view objects in the shared pool that are always present and/or have increasing reload numbers to help identify objects that are good candidates for pinning.

Troubleshooting

If a problem is found in Oracle8i or earlier, do the following:

- Edit the Init.ora file for the database.
- Increase the amount of shared_pool_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor the new value to see if performance improves.

If a problem is found in Oracle9i or later, do the following:

- Increase the size of the shared_pool_size parameter through use of the ALTER SYSTEM SET shared_pool_size command value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

If you determine that SQL literals are causing SQL to not be reused, do the following (in Oracle 8.1.6 and later):

- Change the cursor_sharing parameter to FORCE by using the ALTER SYSTEM SET cursor_sharing=FORCE command.
- Monitor database to see if parse activity is reduced and library cache reloads shrink.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started. If using an Init.ora file, add the cursor_sharing=FORCE parameter to the file.

Dictionary Cache Hit Ratio

Oracle's shared pool offers an number of tuning possibilities and is made up of two main memory areas:

1 Library Cache

2 Data Dictionary Cache

The dictionary cache hit ratio offers another key indicator in determining the performance of the shared pool. It shows how often object definitions are found in memory vs. having to read them in from disk. Because Oracle references the data dictionary many times when an SQL statement is processed, it is imperative that as much of this vital reference information be kept in RAM as possible.

> **NOTE:** This statistic is also available on the Memory home page.

Metrics

Just as with the library cache, a high data dictionary cache hit ratio is desirable. Strive for a hit ratio between 90-100%, with 95% being a good performance benchmark.

> **NOTE:** When a database is first started, the data dictionary cache hit ratio is not at an optimal level because all references to object definitions are relatively new, and as such, must be placed into the shared pool. Look for hit ratios in the eighty's for new database startups. If, however, after a solid hour or two of steady database time, the data dictionary cache hit ratio has not increased to desirable levels, increase the shared_pool_size parameter.

Databases supporting applications that involve large number of objects (such as an Oracle Financials installation) should have larger than normal shared pools to support the required object definitions.

Although each parameter is not individually tunable (it was in Oracle6), you can see which area of the dictionary cache could be pulling the overall hit ratio down.

Troubleshooting

If a problem is found in Oracle8i or earlier, do the following:

- Edit the Init.ora file for the database.
- Increase the amount of shared_pool_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor new value to see if performance improves.

If a problem is found in Oracle9i or later, do the following:

- Increase the size of the shared_pool_size parameter through use of the ALTER SYSTEM SET shared_pool_size command value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

Memory Sort Ratio

Oracle's SGA is not the only memory structure used by Oracle for database work. One of the other memory areas used by Oracle8i and earlier for normal activity is an area set aside for sort actions. When a sort operation occurs, Oracle attempts to perform the sort in a memory space that exists at the operating system level. If the sort is too large to be contained within this space, it continues the sort on disk - specifically, in the user's assigned TEMPORARY TABLESPACE. Oracle records the overall number of sorts that are satisfied in memory as well as those that end up being finalized on disk. Using these numbers, you can calculate the percentage of memory sorts vs. disk sorts and get a feel for how fast your sort activity is being resolved. Obviously, memory sorts completes many times faster than sorts forced to use physical I/O to accomplish the task at hand.

Oracle9i or later now has the option of running automatic PGA memory management. Oracle has introduced a new Oracle parameter called pga_aggregate_target. When the pga_aggregate_target parameter is set and you are using dedicated Oracle connections, Oracle ignores all of the PGA parameters in the Oracle file, including sort_area_size, hash_area_size and sort_area_retained_size. Oracle recommends that the value of pga_aggregate_target be set to the amount of remaining memory (less a 10% overhead for other server tasks) on a server after the instance has been started.

NOTE: This statistic is available on the Home page, Memory home page, and the Users home page.

Metrics

If the memory sort ratio falls below 90%, and you are on Oracle8i or earlier, increase the parameters devoted to memory sorts - sort_area_size and sort_area_retained_size.

For Oracle9i or later, investigate the use of pga_aggregate_target. Once the pga_aggregate_target has been set, Oracle automatically manages PGA memory allocation, based on the individual needs of each Oracle connection. Oracle9i or later allows the pga_aggregate_target parameter to be modified at the instance level with the alter system command, thereby lets you dynamically adjust the total RAM region available to Oracle9i.

Oracle9i also introduced a new parameter called workarea_size_policy. When this parameter is set to automatic, all Oracle connections benefits from the shared PGA memory. When workarea_size_policy is set to manual, connections allocates memory according to the values for the sort_area_size parameter. Under the automatic mode, Oracle tries to maximize the number of work areas that are using optimal memory and uses one-pass memory for the others.

Troubleshooting

If you find a problem, do the following:

- Edit the Init.ora or SPFILE file for the database.
- Increase the amount of sort_area_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value. EVERY user receives this amount for sorting.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor new value to see if performance improves.

In addition to increasing the amount of memory devoted to sorting, find inefficient SQL that cause needless sorts. For example, UNION ALL does not cause a sort whereas UNION does in an SQL query (to eliminate duplicate rows). DISTINCT is frequently misapplied to SQL statements and causes unnecessary sort actions.

There are times you simply cannot stop sort activity. This being the case, try to keep it in memory whenever possible. However, large data warehousing systems oftentimes simply exhaust RAM sort allotments, so if disk sorts must occur, try to ensure three things:

1 Your user's TEMPORARY TABLESPACE assignment is not the SYSTEM tablespace, which is the default assignment.

   NOTE: For Oracle9i or later, you can specify a default tablespace other than SYSTEM for every user account that is created.

2 The TEMPORARY TABLESPACE assigned to your users is placed on a fast disk.

3 The TEMPORARY TABLESPACE has the tablespace parameter TEMPORARY assigned to it, which allows sort activity to be performed in a more efficient manner.

Parse/Execute Ratio

Each time a new SQL statement is submitted to Oracle, the kernel must ‘parse’ the statement, which involves syntax checks, security checks, and object validations. The Parse/Execute Ratio shows the percentage of SQL executed that did not incur a hard parse.

NOTE: This statistic is available on the Home page, Memory home page, and the Users home page.

Metrics

Seeing low values might indicate that users are firing SQL with many hard-coded literals instead of using bind variables within an application. High values (90% and greater) generally indicate Oracle is saving precious CPU by avoiding heavy parse tasks.

Troubleshooting

The best way to reduce unnecessary parse activity is to encourage SQL statement reuse. This can be done by promoting SQL execution through the use of stored procedures or applications where bind variables can be used. Oftentimes, literals in otherwise identical SQL statements can cause unneeded parse work for Oracle. The use of bind variables can counter that problem.

If you determine that SQL literals are causing SQL to not be reused, do the following (in Oracle 8.1.6 and later):

- Change the cursor_sharing parameter to FORCE by using the ALTER SYSTEM SET cursor_sharing=FORCE command.
- Monitor database to see if parse activity is reduced and library cache reloads shrink.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started. If using an Init.ora file, add the cursor_sharing=FORCE parameter to the file.

Buffer Busy Waits

Buffer busy waits occur when a process needs to access a data block in the buffer cache, but cannot because it is being used by another process. So it must wait. A wait event generally happens because a buffer is being read into the buffer cache by another process or the buffer is in the buffer cache, but cannot be switched to a compatible mode immediately.

Metrics

Buffer busy waits normally center around contention for rollback segments, too small an INITRANS setting for tables, or insufficient free lists for tables.

Troubleshooting

On Oracle8i or earlier, the remedy for each situation would be increasing the number of rollback segments, or altering tables to have larger settings for INITRANS to allow for more transactions per data block, and more free lists.

For Oracle9i or later, you can use the automatic segment management feature in Oracle9i locally-managed tablespaces to help make free list problems a thing of the past. Using an UNDO tablespace in 9i or later can help remedy any rollback contention problem.

You can also obtain which objects have actually experienced buffer busy waits in Oracle9i or later by querying the sys.v_$segment_statistics. This view is not populated unless the configuration parameter statistics_level is set to TYPICAL or ALL.

Rollback Contention Ratio

Rollback segments are used by Oracle to hold data needed to rollback (or undo) any changes made through inserts, updates, or deletes to various Oracle objects. They also allow Oracle to have read consistency for long running queries, are used for recovery purposes, and play a role during exports of database information. In a heavy transaction processing environment, rollback segments are accessed continuously and therefore are subject to contention problems. The Rollback Contention Ratio helps identify contention occurring on the system relating to rollbacks.

Metrics

Overall, if the rollback contention ratio approaches 1% or more, create more rollback segments. Also consider creating a specialized, larger, rollback segment to be used by long running transactions. Doing so alleviates dynamic rollback extensions and cuts down heavily on ORA-01555 Snapshot Too Old errors.

Troubleshooting

Begin by creating new rollback segments and altering them to be online for use. Then monitor the overall contention ratio to see if it begins to drop.

If you are using Oracle9i or later, consider using an UNDO tablespace and allowing Oracle to automatically control rollback segment management.

Latch Miss Ratio

Protecting the many memory structures in Oracle's SGA are latches. They ensure that one and only one process at a time can run or modify any memory structure at the same instant. Much more restrictive than locks (which at least allow for some collective user interaction), latches have no queuing mechanism - so either you get it or you do not and are forced to continually retry.

The latch miss ratio defines the number of times a process obtained a willing-to-wait latch vs. missing the attempt.

Metrics

If the latch miss ratio exceeds 1%, you should take action to resolve the amount of latch contention.

Troubleshooting

Examine the details regarding the latch contention. Increasing the shared_pool_size can assist in latch problems also. The table below describes latches:

Latch	Description
Cache buffer chain latch	Protects paths to database block buffers in the buffer cache. High I/O loads tend to cause contention for this latch. You can alleviate contention somewhat by adding more buffers to the cache (through the db_block_buffers or db_cache_size parameter) or by adding more LRU latch chain latches with the db_block_lru_latches parameter.
Library cache latches	Protects cached SQL statements in the library cache area of the Oracle shared pool. Contention for this latch is the usual result of literals being used for SQL statements instead of bind variables.

Other routine latch contention problems used to include the redo allocation and redo copy latches, but these have pretty much been made obsolete in Oracle 8.1.5 and later.

Parallel Query Busy Ratio

Oracle's parallel query feature, when used properly, allows for terrific increases in performance for many SQL and utility operations. Parallel operations can be introduced through SQL hints, specified in an object's DDL, or used in command line arguments for utility programs (like SQL*Loader). To effectively service parallel query requests, ensure that enough query servers exist in the database instance. The Parallel Query Busy Ratio is an indicator of how busy all the servers are on the database in question.

Metrics

If the Parallel Query Busy Ratio approaches 80-90%, add more query servers to the database, or examining parallel requests to ensure they are being used in an efficient and necessary manner.

Troubleshooting

To fix, do the following:

- Edit the Init.ora file or SPFILE for the database.
- Increase the amount of parallel_max_servers to a higher value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor new value to see if performance improves.

You can also investigate the use of the parallel_automatic_tuning parameter in Oracle 8.1 and later.

Free Shared Pool Percent

Oracle's shared pool need not use all of the memory given to it through the shared_pool_size parameter. If the database does not have many object and code definitions to reference, then the shared pool can contain an abundance of free memory that is not being used.

Metrics

Under-allocating the shared pool size can have a serious impact on your database's performance, but over-allocating the shared pool can have run time ramifications as well. If you have a good chunk of memory allocated to the Oracle shared pool that is never used, it might be more of a performance enhancement to reduce the shared pool amount and instead give the memory to the buffer/data cache, or even back to the operating system itself. In terms of knowing when to reduce the shared pool, a good benchmark is continually seeing 2-3MB of free memory.

On the other hand, if after an hour or so of beginning database operation, you see that virtually no free memory is left in the shared pool, or you are seeing ORA-4031 errors (that indicate definitions cannot find enough contiguous free space in the shared pool), increase the pool by 10% or more.

Troubleshooting

If you continuously see little or no free memory in the shared pool, do the following:

For Oracle8i or earlier:

- Edit the Init.ora file for the database.
- Increase the amount of shared_pool_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor the new value to see if performance improves.

For Oracle9i or later:

- Increase the size of the shared_pool_size parameter through use of the ALTER SYSTEM SET shared_pool_size command value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

You can also use the ALTER SYSTEM FLUSHED SHARED_POOL command to remove all objects from the shared pool and start with a clean slate.

Problem Tablespaces

The Problem Tablespaces metric is a count of all tablespaces in the database whose free space percentage amount has fallen below a user-defined limit.

Metrics

A rule of thumb for dynamic tablespaces (those with growing objects) is to keep a minimum of 10-15% free space available for object growth.

Troubleshooting

There are two ways to prevent a tablespace from running out of available free space:

- Turn AUTOEXTEND on for the underlying tablespace's datafiles. This allows them to automatically grow when free space in the datafile has been exhausted.
- Using the ALTER TABLESPACE … ADD DATAFILE… command, you can manually add a new datafile to a tablespace that is about to run out of available free space.

Problem Objects

The Problem Objects statistic is a count of all objects in the database that are in danger of reaching their maximum extent limit or cannot allocate their next extent of free space because of a lack of overall or contiguous free space in their parent tablespace.

Metrics

Modify any object approaching their maximum extent limit or unable to allocate a new extent of space so they can continue to grow in size.

Troubleshooting

Depending on the situation, there are a number of things you can do to prevent object extent problems:

- Turn AUTOEXTEND on for the underlying parent tablespace's datafiles. This allows a tablespace to automatically grow when free space in the datafile has been exhausted, and allows an object to extend even when little or no current free space is available.
- Using the ALTER TABLESPACE … ADD DATAFILE… command, you can manually add a new datafile to a tablespace that is about to run out of available free space.
- You can alter an object that is at or near their maximum extent limit so that the object has unlimited extents.
- With Oracle 8.1.5 and later, you can use locally-managed tablespaces to ensure that no object ever reaches its maximum extent limit, because all objects are allowed unlimited extents.
- An object can be reorganized into another tablespace or reorganized in general to reduce the number of extents the object currently takes up.

Top System Bottlenecks

When viewing wait statistics, there are many levels of detail that you can view. The first level is the system view, which provides a global, cumulative snapshot of all the waits that have occurred on a system. Viewing these numbers can help you determine which wait events have caused the most commotion in a database thus far. The Top System Bottlenecks section identifies the top waits that have occurred on the Oracle database based on the number of waits per event.

Metrics

None.

Troubleshooting

Appendix A in the Oracle Reference manual contains a listing and description of every current wait event defined in Oracle. DBAs unfamiliar with what each event represents should keep this listing close by as they examine wait-based event metrics. For example, a ‘db file scattered read’ event is typically indicative of table scan operations. If you see many of these events, then you can begin to see if large table scans are occurring in the database. Like the ‘db file scattered read’ event, each wait event has its own meaning and individual end-resolution diagnosis.

After looking at system-level wait activity, you can discover which current connections are responsible for waits at the system level. Performance Analyst reports on historical and current wait events at the session level, making this investigation easy to accomplish.

Top Session Bottlenecks

When viewing wait statistics, there are many levels of detail that you can view. The first level is the system view, which provides a global, cumulative snapshot of all the waits that have occurred on a system. The second level is a historical look at waits from a session level. The third is which sessions are currently experiencing waits. The Top Session Bottlenecks section identifies the top sessions that are currently waiting, based on their wait time in seconds.

Metrics

None.

Troubleshooting

Appendix A in the Oracle Reference manual contains a listing and description of every current wait event defined in Oracle. DBAs unfamiliar with what each event represents should keep this listing close by as they examine wait-based event metrics.

The most common wait viewed at the session level is an ‘enqueue’ wait, which typically identifies lock contention. If enqueue waits are observed, then you can check the “Current Object Blocks” count on the Performance Analyst Home page, as well as the Users page which displays locks and blocking locks detail.

As with an enqueue event, each wait event has its own meaning and individual end-resolution diagnosis.

Current Object Blocks

A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches even on large systems. Although Oracle supports unlimited row-level locking, blocking lock situations do crop up. Blocks are most often caused by user processes holding exclusive locks and not releasing them via a proper COMMIT frequency.

> **NOTE:** This statistic is also called Sessions Blocked on the Users home page and Session Blocks on the Objects home page.

Metrics

Investigate any indicator above zero immediately before the situation has a chance to mushroom.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied. You can issue a KILL against the offending process, which eliminates the user's stranglehold on the objects they were accessing. Other user processes then nearly almost always complete in an instant. Discovering the blocked lock situation is made easier by using tools like Performance Analyst, but preventing the blocking lock situation in the first place is where it gets tricky. You can look at the Users Detail and view all current blocking locks to see exactly which sessions are holding the currently restrictive locks.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. Most DBAs who have had to face Oracle Forms applications have suffered through the dreaded SELECT … FOR UPDATE statements that place unnecessary restrictive locks on nearly every read operation, and know all too well that good coding practice is important. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

Enqueue Waits

An enqueue is an advanced locking device that allows multiple database processes to share certain resources. Enqueue waits typically occur when sessions wait to be granted a requested lock. Sometimes these locks are internal Oracle locks while other times they could be locks for rows of data in a table. Enqueues are issued implicitly by Oracle.

NOTE: This statistic is available on the Home page and the Objects home page.

Metrics

Investigate any enqueue waits that read consistently above one or more (delta statistics).

Troubleshooting

Removing contention for enqueues is almost always an application design issue. If many enqueue waits are seen, this normally indicates either contention for certain rows in the database, or the result of database-initiated lock escalation. Examine the use of indexes to make sure all referencing foreign keys are indexes and that SQL is tuned to not tarry over rows in the database during modification operations.

Enqueue waits can also be the result of space management tasks (such as objects extending) and disk sorts (mainly in tablespaces that do not make use of the TEMPORARY tablespace parameter).

Free List Waits

Free lists are lists of Oracle data blocks that contain free space for an Oracle object. Every table has at least one free list. Free lists are used to locate free blocks of space when a request is made of a table for the insertion of a row. Free list contention can reduce the performance of applications when many processes are involved in the insertion of data to the same table.

NOTE: This statistic is available on the Home page and the Objects home page.

Metrics

If consistent numbers for free list waits continues to appear, add additional free lists to the most dynamic database objects through the use of the STORAGE parameter. Another indicator of insufficient free lists are consistent, non-zero numbers for the buffer busy wait event.

Troubleshooting

If free list waits are observed, add more free lists to tables and indexes with high insert rates. For Oracle9i or later, objects can be created inside of locally-managed tablespaces that use the automatic segment management feature, which entirely does away with free lists.

Total Used Space/Total Free Space

These statistics represent the total used and free space available in all tablespaces/datafiles in the database. Although good to know, a more detailed listing by tablespace is needed to determine where any actual space shortages exist in the database.

NOTE: These statistics are in the Storage Analysis section of the Performance Analyst Home page and the Space home page.

Metrics

If any one tablespace begins to approach 90% used (and the objects contained within it are dynamic and growing as opposed to static), take action to prevent any future space allocation errors.

Troubleshooting

Here are a some things you can do to prevent a tablespace from running out of available free space:

- Turn AUTOEXTEND on for the underlying tablespace's datafiles. This allows them to automatically grow when free space in the datafile has been exhausted.
- Using the ALTER TABLESPACE … ADD DATAFILE… command, you can manually add a new datafile to a tablespace that is about to run out of available free space.

For more information, see Storage Analysis.

Archive Log

Oracle can be placed into archivelog mode, which tells the database to make copies of the online redo log files for point-in-time recovery purposes. The Archive Log statistic displays whether the database is running in archivelog mode or not. This information is displayed in the Storage Analysis section of the Performance Analyst Home page.

Metrics

None.

For more information, see Storage Analysis.

Top Processes

When the database population as a whole experiences a system slowdown, it is not uncommon to find one or two users who are responsible for bringing the system to its knees. In the best of worlds, users have an evenly divided amount of memory usage, disk I/O, CPU utilization, and parse activity. Unfortunately, this is not usually the case. Users submit large batch jobs during peak OLTP activity, or when sessions that are firing off untuned queries on a critical system.

If you are seeing a slowdown in your database, and cannot seem to find a root cause, one thing to examine is the resource consumption of the leading sessions on a system. A handful of users can overwhelm the I/O capabilities of Oracle (through untuned queries or runaway batch jobs) or hammer the CPU or memory structures of the database and server.

Performance Analyst makes it easy to pinpoint the top sessions by showing the leading processes at the physical I/O, logical I/O, memory, and CPU usage levels.

Metrics

If any one session uses more than 50% of a total resource (CPU, memory, etc.) go into the session to find out what they are currently executing.

Active User Processes

The Active User Processes statistic is the total number of active and open threads reported in the database. Active Sessions displays the number of processes actively performing work.

Metrics

None.

For more information, see Inactive User Processes.

Inactive User Processes

The Inactive User Processes statistic is the total number of threads logged on to the database that are idle at the current time.

Metrics

A large number of inactive users could indicate user sessions that have mistakenly been left logged on. Because each user thread consumes a portion of memory on the Oracle server, sever any sessions not needing a connection to reduce resource usage.

Troubleshooting

Double-click this statistic to open the Top Sessions tab of the Users Detail page. On this tab you can check sessions that have many seconds idle and/or that have been logged on for very long periods of time as indicated by the logon time column. After verifying that the session is no longer necessary, you can KILL the session.

For more information, see Active User Processes.

Memory Page Statistics

The Memory home page includes the following sections:

- Bottleneck Analysis
- Key Ratio Analysis
- SGA Analysis
- SQL Analysis
- Workload Analysis - Top Memory Hogs
- Buffer Cache Hit Ratio
- Buffer Pool Allocations
- Buffer Pool Usage
- Data Dictionary Cache Hit Ratio
- Free Memory in Shared Pool
- Library Cache Hit Ratio
- Memory Sort Ratio
- Disk Sort Ratio
- Session Leaders - Memory
- SGA

Related Topics

Memory Detail

Home Page Statistics

I/O Page Statistics

Objects Page Statistics

OS Page Statistics

Space Page Statistics

Top SQL Page Statistics

Users Page Statistics

Network Statistics

Contention Statistics

Key Ratio Analysis - Memory

Object-related database activity can be examined using both ratio-based and wait/bottleneck-based analysis. Ratio-based analysis involves examining a number of key database ratios and statistical readings that can be used to indicate how active certain object types are. Performance ratios serve as roll-up mechanisms for busy DBAs to use for at-a-glance performance analysis.

When using ratio-based analysis, there are some standards to adhere to. To start with, many of the formulas that make up ratio-based analysis must be derived from delta measurements instead of cumulative statistics. Many of the global ratios that you examines come from the v$sysstat performance view. The performance view maintains a count of all the occurrences (in the VALUE column) of a particular database incident (in the NAME column) since the database was brought up. For databases that are kept up for long periods of time, these values can grow quite large and impacts how a particular ratio is interpreted. However, if delta statistics are used (taking, for a specified sampling period, the before and after counts of each statistic that make up a ratio), then an accurate and current portrayal of the various ratios can be had.

A final thing to remember about using ratio-based analysis is that, while there are a number of rules of thumb that can be used as starting points to begin the evaluation of database performance, DBAs must determine each database's individual 'personality' with respect to the key performance ratios. Some hard and fast rules simply do not apply to every database. The danger in using blanket ratio standards is that they can lead you to haphazardly take action, which can at times contribute nothing to the situation, and sometimes even degrade performance.

The following memory ratios are used on the Performance Analyst Memory home page to succinctly communicate the general overall memory performance levels of the monitored database:

- Buffer Cache Hit Ratio
- Library Cache Hit Ratio
- Dictionary Cache Hit Ratio
- Memory Sort Ratio
- Parse/Execute Ratio
- Free Shared Pool Percent

For more information, see Memory Page Statistics.

Bottleneck Analysis - Memory

When an Oracle database is up and running, every connected process is either busy doing work or waiting to perform work. A process that is waiting may mean nothing in the overall scheme of things or it can be an indicator that a database bottleneck exists. You can use Bottleneck Analysis to determine if perceived bottlenecks in a database are contributing to a performance problem.

Memory bottlenecks can definitely cause performance degradation in an otherwise well-running database. Typically, these bottlenecks center around Oracle's buffer/data cache, library cache, and occasionally log buffer memory regions. To help you identify such problems, the following statistics are presented on the Memory home page:

- Buffer Busy Waits
- Free Buffer Wait Average
- Object Reloads
- Redo Log Space Wait Time
- Top Latch Misses

For more information, see:

Memory Home Page

Buffer Busy Ratio

SQL Analysis - Memory

Much of a database's overall performance can be attributed to SQL statement execution. Poorly optimized SQL statements can drag an otherwise well-configured database down in terms of end user response times. SQL statements that use much memory can also cause a problem in a database. The SQL Analysis for memory shows what SQL statements have consumed the largest percentages of shareable, persistent, and runtime memory

Before you can identify problem SQL in your database, you have to ask the question, "What is bad SQL?" What criteria do you use when you begin the hunt for problem SQL in your critical systems? Understand that even the seasoned experts disagree on what constitutes efficient and inefficient SQL; so there is no way to sufficiently answer this question to every Oracle professional's satisfaction. The table lists some general criteria you can use when evaluating the output from various database monitors or personal diagnostic scripts:

Criteria	Description
Overall Response (Elapsed) Time	The time the query took to parse, execute, and fetch the data needed to satisfy the query. It should not include the network time needed to make the round trip from the requesting client workstation to the database server. This statistic is available in Oracle9i or later.
CPU Time	The CPU time the query took to parse, execute, and fetch the data needed to satisfy the query.
Physical I/O	This is often used as the major statistic in terms of identifying good vs. bad SQL, this is a measure of how many disk reads the query caused to satisfy the user's request. While you certainly want to control disk I/O where possible, it is important that you not focus solely on physical I/O as the single benchmark of inefficient SQL. Make no mistake, disk access is slower than memory access and also consumes processing time making the physical to logical transition, but you need to look at the entire I/O picture of a SQL statement, which includes looking at a statements' logical I/O as well.
Logical I/O	The number of memory reads the query took to satisfy the user's request. The goal of tuning I/O for a query should be to examine both logical and physical I/O, and use appropriate mechanisms to keep both to a minimum.
Repetition	The number of times the query has been executed. A problem in this area is not as easy to spot as the others unless you know your application well. A query that takes a fraction of a second to execute can be a headache on your system if it has executed erroneously (for example, a query that executes in a runaway PL/SQL loop) over and over again.

There are other criteria that you can examine like sort activity or access plan statistics (that show items like Cartesian joins and the like), but more often than not, these measures are reflected in the criteria listed above.

Fortunately, Oracle records all the above measures (some only in 9i), which makes tracking the SQL that has been submitted against an Oracle database much easier.

Metrics

When you begin to look for inefficient SQL in a database, there are two primary questions you want answered:

1 What HAS been the worst SQL that has historically been run in my database?

2 What IS the worst SQL that is running right now in my database?

When troubleshooting a slow system, you should be on the lookout for any query that shows an execution count that is significantly larger than any other query on the system. It could be that the query is in an inefficient PL/SQL loop, or other problematic programming construct. Only by bringing the query to the attention of the application developers will you know if the query is being mishandled from a programming standpoint.

There is the possibility that the SQL statement just is not tuned well. You can view Performance Analyst's Top SQL view and, if you have Embarcadero SQL Tuner installed, you can port the SQL over to SQL Tuner to better optimize the statement.

SGA Analysis - Memory

Most DBAs know all about the Oracle System Global Area (SGA). The SGA is Oracle's memory structural area devoted to facilitating the transfer of data and information between clients and the Oracle database. The table below describes Oracle memory structures:

Memory Structure	Description
Default buffer cache	Maintains data blocks when they are read from the database. If you do not specifically place objects in another data cache, then any data requested by clients from the database is placed into this cache. The memory area is controlled by the db_block_buffers parameter in Oracle8i and earlier and db_cache_size in Oracle9i or later.
Keep buffer cache	For Oracle 8 or later, you can assign various objects to a special cache that retains those object’s requested blocks in RAM for as long as the database remains up. The keep cache's main is for often-referenced lookup tables that should be kept in memory at all times for fast access. The buffer_pool_keep parameter controls the size of this cache in Oracle8, while the db_keep_cache_size parameter handles the cache in Oracle9i or later. The keep pool is a sub-pool of the default buffer cache.
Recycle buffer cache	The opposite of the keep cache. When large table scans occur, the data that fills a memory cache will likely not be needed again and should be quickly discarded from RAM so that they do not occupy memory space and prevent needed blocks from assuming their place. Objects containing such data can be assigned to the recycle pool to ensure that such a thing does indeed occur. The buffer_pool_recycle parameter controls the size of this cache in Oracle8 and earlier, while the db_recycle_cache_size parameter handles the cache in Oracle9i or later.
Specific block size caches	For Oracle 8 or later, you can create tablespaces whose blocksize differs from the overall database blocksize. When data is read into the SGA from these tablespaces, their data has to be placed into memory regions that can accommodate their special block size. Oracle9i or later has memory settings for 2K, 4K, 8K, 16K, and 32K caches. The configuration parameter names are in the pattern of db_nk_cache_size.
Shared pool	Holds object structure as well as code definitions, and other metadata. Setting the proper amount of memory in the shared pool assists a great deal in improving overall performance with respect to code execution and object references. The shared_pool_size parameter controls the memory region.

Memory Structure	Description
Large pool	For Oracle 8 or later, you can configure an optional, specialized memory region called the large pool that holds items for shared server operations, backup and restore tasks, and other miscellaneous things. The large_pool_size parameter controls the memory region. The large pool is also used for sorting when the multi-threaded server (MTS) is implemented.
Java pool	Handles the memory for Java methods, class definitions, etc. The java_pool_size parameter controls the amount of memory for this area.
Redo log buffer	Buffers modifications that are made to the database before they are physically written to the redo log files. The log_buffer configuration parameter controls the memory area.

Oracle also maintains a fixed area in the SGA that contains a number of atomic variables, pointers, and other miscellaneous structures that reference areas of the SGA.

For more information, see Memory Home Page.

Workload Analysis - Top Memory Hogs

It is not uncommon for one or two users to cause the majority of runtime problems that plague a database. The problem could be a runaway process, an untuned batch procedure, or other user-initiated operation. Oftentimes, user connections can get out of hand with memory consumption, and extreme cases have caused headaches at both the database and operating system level (ORA-4030 errors).

If your database server does not have an overabundance of memory, periodically check to see who your heavy memory users are along with the total percentage of memory each takes up. If you see one or two users who have more than 25-50% of the total memory usage, investigate the sessions further to see the activities they are performing.

For more information, see Memory Home Page.

Buffer Cache Hit Ratio

- Metrics
- Troubleshooting

The buffer cache hit ratio is an indicator of how often user requests for data are satisfied through memory vs. being physically read from disk. Data read from memory produces user response times many times faster than when that same data is read from disk. Keeping physical I/Os to an absolute minimum is one of the Oracle buffer cache's purposes in life.

The table below describes the key counters Performance Center uses to calculate the buffer cache hit ratio:

Key Counter	Description
DB BLOCK GETS	The number of times a CURRENT block was requested. You review values for this statistic in two places. First, go to Memory Detail>Session Detail>Session I/O. Second, you can add this statistic to the Trends view.
CONSISTENT GETS	Data read from rollback segments in memory.
PHYSICAL READS	Data read physically from disk.
Direct Reads	Data read physically from disk that bypasses the buffer cache. Direct reads are filtered out of overall physical reads so an accurate cache hit ratio can be determined.

Dividing the data read from memory by data read from disk yields the cache hit ratio.

TIP: Click this statistic to drill down to the Hit Ratio by User tab of the Memory Detail view.

Metrics

To help ensure excellent performance, you want to keep your cache hit ratio in the neighborhood of 90% or higher. However, every database has its own 'personality' and can exhibit excellent performance with below average readings for the cache hit ratio. Excessive logical I/O activity can produce a very high cache hit ratio while actually degrading overall database performance.

You should investigate consistent low readings of 60% or lower.

NOTE: You must cycle the Oracle server to allow any increases in db_block_buffers to take effect.

NOTE: For Oracle8i or lower, the adjustment of the `db_block_buffers` tuning parameter is required. For Oracle9i or higher, the `db_cache_size` parameter is the parameter that needs attention. Any increases in `db_block_buffers` to take effect on Oracle8i or lower, the database must be cycled. The `db_cache_size` parameter in Oracle9i or later, however, is dynamic and can be altered without stopping and starting the database instance.

To view individual session hit ratios that can be depressing the overall cache hit ratio for the database, drill down into the overall buffer cache hit ratio.

Troubleshooting

If a problem is found in Oracle8i or earlier:

- Edit the Init.ora file for the database.
- Increase the amount of db_block_buffers to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor the new value to see if performance improves.

If a problem is found in Oracle9i or later:

- Increase the size of the db_cache_size parameter through use of the ALTER SYSTEM SET db_cache_size command value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

Buffer Pool Usage

Buffer cache usage statistics show the state and use of the buffers currently in the buffer cache and they include a count of how many there are for each state. Typically, the statuses of the buffers are:

- Free
- Read and modified

- Read and not modified
- Currently being read

Metrics

This grouping of statistics is quite helpful in determining if you have an overabundance of block buffers allocated. Consistently seeing large number of free buffers clues you in to the fact that you can reduce the amount of block buffers in the cache and give memory back to the operating system.

If, however, you see no free buffers within the first hour of bringing your Oracle database up, you can consider adding more buffers to the cache.

Troubleshooting

If you find a problem, do the following:

1. Edit the Init.ora file for the database.
2. Increase the amount of db_block_buffers to a higher value if free buffers are not found (take care not to over-allocate; ensure enough free memory exists on server before increasing value). Reduce the number of an overabundance of FREE buffers are present.
3. Cycle the Oracle server when possible to allow the new value to take effect.
4. Monitor the new value to see if more numbers of free buffers show up for the FREE status.

Buffer Pool Allocations

Because you can reference data objects with different usage patterns, Oracle8 offers the option to intelligently place objects into one of three buffer caches:

Buffer Cache	Description
KEEP	Minimizes misses in the buffer cache. Small, frequently referenced objects are candidates for the KEEP buffer pool.
RECYCLE	Lets you avoid having large numbers of infrequently accessed blocks of data crowd out objects that need to be referenced in RAM. The RECYCLE pool is good for large objects that are scanned from beginning to end without the need for keeping all their data in RAM.
DEFAULT	The traditional cache for all other data objects.

NOTE: Unless you specify the KEEP or RECYCLE buffer cache, Oracle automatically places objects into the DEFAULT buffer cache.

TIP: Click this statistic to drill down to the Buffer Pool tab of the Memory Detail view.

Metrics

When looking at overall database I/O activity, you should keep an eye out for objects that can lend themselves to placement into a particular buffer cache. Consider using a KEEP cache for relatively small, frequently accessed tables that require fast response times. Large tables with random I/O activity and are scanned from beginning to end a lot are good candidates for a RECYCLE cache.

Troubleshooting

You can use the STORAGE...BUFFER _POOL option to place objects into different buffer pools at object creation time. You use the ALTER command to set existing objects different pool.

> **TIP:** Tables, partitions, and indexes can be placed into the different caches.

If you just want to use the DEFAULT buffer pool and not enable any special caches, you can still encourage Oracle to keep certain objects in the buffer cache as long as possible by using the CACHE parameter. For example, issuing the command ALTER TABLE…CACHE specifies that the blocks retrieved for this table be placed at the most recently used end of the LRU list in the DEFAULT buffer cache when a full table scan is performed. The CACHE hint can also be used in SQL statements to cache a table, but used in this form, Oracle only caches the blocks until the next time the database is shut down. Once the database comes back up, the CACHE hint would have to be issued in an SQL statement again to cache the needed blocks of data.

Free Memory and Used Memory in Shared Pool

Oracle's shared pool does not need to use all of the memory provided through the shared_pool_size parameter. If the database does not have many object and code definitions to reference, the shared pool can contain an abundance of unused free memory. This statistic is tracked by Oracle in its performance views.

> **TIP:** Click this statistic to drill down to the Shared Pool tab of the Memory Detail view.

Metrics

Under-allocating the shared pool size can have a serious impact on your database's performance and over-allocating the shared pool can have runtime ramifications. If you have a good chunk of memory allocated to the Oracle shared pool that is never used, it might be more of a performance enhancement to reduce the shared pool amount and instead give the memory to the buffer cache, or even back to the operating system itself. In terms of knowing when to reduce the shared pool, a good benchmark is continually seeing 2-3 megabytes (MB) of free memory.

On the other hand, if after an hour or so of beginning database operation, you see virtually no free memory is left in the shared pool, or you are seeing ORA-4031 errors (that indicate definitions cannot find enough contiguous free space in the shared pool), you should definitely look into increasing the pool by 10% or more.

Troubleshooting

If you continuously see little or no free memory in the shared pool then you can do the following on Oracle8i or lower:

- Edit the init.ora file for the database.
- Increase the amount of `shared_pool_size` to a higher value (take caution to not over-allocate; ensure enough free memory exists on server before increasing value.)
- Cycle the Oracle server when possible to allow the new value to take affect.
- Monitor the new value to see if performance improves.

If a problem is found in Oracle9i or later:

- Increase the size of the `shared_pool_size` parameter through use of the `ALTER SYSTEM SET shared_pool_size` command value (take caution to not over-allocate; ensure enough free memory exists on server before increasing value.)
- Monitor the new value to see if performance improves.

- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

You can also use the `ALTER SYSTEM FLUSHED SHARED_POOL` command to remove all objects from the shared pool and start with a clean slate.

Library Cache Hit Ratio

- Metrics
- Troubleshooting

Oracle's shared pool offers a number of tuning possibilities and is made up of two main memory areas:

1 Library Cache

2 Data Dictionary Cache

The library cache hit ratio offers a key indicator in determining the performance of the shared pool. It holds commonly used SQL statements - basically database code objects. The library cache hit ratio shows how often SQL code is reused by other database users vs. the number of times an SQL statement is broken down, parsed, and then loaded (or reloaded) into the shared pool.

You can improve performance by encouraging the reuse of SQL statements so expensive parse operations can be avoided. The library cache assists this tuning effort.

NOTE: This statistic is also available on the Home page, the Memory home page and on the Library Cache tab of the Memory Detail.

TIP: Click this statistic to drill down to the Library Cache tab of the Memory Detail view.

Metrics

A high library cache hit ratio is a desirable thing. Strive for a hit ratio between 95-100%, with 99% being a good performance benchmark for code reuse.

NOTE: When a database is first started, the library cache hit ratio is not at an optimal level because all code being used is relatively new, and as such, must be parsed and placed into the shared pool. If, however, after a solid hour or two of steady database time, the library cache hit ratio has not increased to desirable levels, increase the shared_pool_size parameter.

Other red flags that can indicate a too small shared pool include:

- A wait count for the event 'latch free' of 10 or greater.
- The library cache wait count of two or greater.

You can track these indicators in Performance Center's Contention performance category view.

A way of improving the library cache hit ratio is by encouraging code reuse through the implementation of bind variables. Discouraging hard coded literals in application code and instead making use of variables bound at run time aids in the reuse of SQL code that is maintained in Oracle's shared pool.

NOTE: Bind variables can have an affect on the cost-based optimizer though.

A second way is to pin frequently used code objects in memory so they are available when needed, by using the system supplied DBMS_SHARED_POOL package. You can use Performance Center to view objects in the shared pool that are always present and/or have increasing reload numbers to help identify objects that are good candidates for pinning.

You can use Performance Center to view objects in the shared pool that are always present and/or have increasing reload numbers to help identify objects that are good candidates for pinning.

Troubleshooting

If a problem is found in Oracle8i or earlier:

- Edit the Init.ora file for the database.
- Increase the amount of shared_pool_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor the new value to see if performance improves.

If a problem is found in Oracle9i or later:

- Increase the size of the shared_pool_size parameter through use of the ALTER SYSTEM SET shared_pool_size command value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

Dictionary Cache Hit Ratio

Oracle's shared pool offers a number of tuning possibilities and is made up of two main memory areas:

1 Library Cache

2 Data Dictionary Cache

The dictionary cache hit ratio offers another key indicator in determining the performance of the shared pool. It shows how often object definitions are found in memory vs. having to read them in from disk. Because Oracle references the data dictionary many times when an SQL statement is processed, it is imperative that as much of this vital reference information be kept in RAM as possible.

TIP: Click this statistic to drill down to the Data Dictionary tab of the Memory Detail view.

Metrics

Just as with the library cache, a high data dictionary cache hit ratio is desirable. Strive for a hit ratio between 90-100%, with 95% being a good performance benchmark.

NOTE: When a database is first started, the data dictionary cache hit ratio is not at an optimal level because all references to object definitions are relatively new, and as such, must be placed into the shared pool. Look for hit ratios in the eighty's for new database startups. If, however, after a solid hour or two of steady database time, the data dictionary cache hit ratio has not increased to desirable levels, increase the shared_pool_size parameter.

Databases supporting applications that involve large number of objects (such as an Oracle Financials installation) should have larger than normal shared pools to support the required object definitions.

Although each parameter is not individually tunable (it used to be in Oracle6), you can see which area of the dictionary cache could be pulling the overall hit ratio down.

Troubleshooting

If a problem is found in Oracle8i or earlier:

- Edit the Init.ora file for the database.
- Increase the amount of shared_pool_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor new value to see if performance improves.

If a problem is found in Oracle9i or later:

- Increase the size of the shared_pool_size parameter through use of the ALTER SYSTEM SET shared_pool_size command value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

Memory Sort Ratio

Oracle's SGA is not the only memory structure used by Oracle for database work. One of the other memory areas used by Oracle8i and earlier for normal activity is an area set aside for sort actions. When a sort operation occurs, Oracle attempts to perform the sort in a memory space that exists at the operating system level. If the sort is too large to be contained within this space, it continues the sort on disk - specifically, in the user's assigned TEMPORARY TABLESPACE. Oracle records the overall number of sorts that are satisfied in memory as well as those that end up being finalized on disk. Using these numbers, you can calculate the percentage of memory sorts vs. disk sorts and get a feel for how fast your sort activity is being resolved. Obviously, memory sorts completes many times faster than sorts forced to use physical I/O to accomplish the task at hand.

Oracle9i or later now has the option of running automatic PGA memory management. Oracle has introduced a new Oracle parameter called pga_aggregate_target. When the pga_aggregate_target parameter is set and you are using dedicated Oracle connections, Oracle ignores all of the PGA parameters in the Oracle file, including sort_area_size, hash_area_size and sort_area_retained_size. Oracle recommends that the value of pga_aggregate_target be set to the amount of remaining memory (less a 10% overhead for other server tasks) on a server after the instance has been started.

> **TIP:** Double-click these statistics to drill down to the Disk Sort Detail tab of the Users Detail view.

Metrics

If your memory sort ratio falls below 90%, and you are on Oracle8i or earlier, increase the parameters devoted to memory sorts - sort_area_size and sort_area_retained_size.

For Oracle9i or later, investigate the use of pga_aggregate_target. Once the pga_aggregate_target has been set, Oracle automatically manages PGA memory allocation, based upon the individual needs of each Oracle connection. Oracle9i or later allows the pga_aggregate_target parameter to be modified at the instance level with the alter system command, thereby lets you dynamically adjust the total RAM region available to Oracle9i.

Oracle9i also introduces a new parameter called workarea_size_policy. When this parameter is set to automatic, all Oracle connections benefits from the shared PGA memory. When workarea_size_policy is set to manual, connections allocates memory according to the values for the sort_area_size parameter. Under the automatic mode, Oracle tries to maximize the number of work areas that are using optimal memory and uses one-pass memory for the others.

Troubleshooting

If you find a problem, do the following:

- Edit the Init.ora or SPFILE file for the database.
- Increase the amount of sort_area_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value. EVERY user receives this amount for sorting.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor new value to see if performance improves.

In addition to increasing the amount of memory devoted to sorting, find inefficient SQL that cause needless sorts. For example, UNION ALL does not cause a sort whereas UNION does in an SQL query (to eliminate duplicate rows). DISTINCT oftentimes is misapplied to SQL statements and causes unnecessary sort actions.

There are times you simply cannot stop sort activity. This being the case, try to keep it in memory whenever possible. However, large data warehousing systems oftentimes simply exhaust RAM sort allotments, so if disk sorts must occur, try to ensure three things:

- Your user's TEMPORARY TABLESPACE assignment is not the SYSTEM tablespace, which is the default assignment.

  NOTE: For Oracle9i or later, you can specify a default tablespace other than SYSTEM for every user account that is created.

- The TEMPORARY TABLESPACE assigned to your users is placed on a fast disk.
- The TEMPORARY TABLESPACE has the tablespace parameter TEMPORARY assigned to it, which allows sort activity to be performed in a more efficient manner.

Free Memory and Used Memory in Shared Pool

Oracle's shared pool need not use all of the memory given to it through the shared_pool_size parameter. If the database does not have many object and code definitions to reference, then the shared pool can contain an abundance of free memory that is not being used.

TIP: Click this statistic to drill down to the Shared Pool tab of the Memory Detail view.

Metrics

Under-allocating the shared pool size can have a serious impact on your database's performance, but over-allocating the shared pool can have run time ramifications as well. If you have a good chunk of memory allocated to the Oracle shared pool that is never used, it might be more of a performance enhancement to reduce the shared pool amount and instead give the memory to the buffer/data cache, or even back to the operating system itself. In terms of knowing when to reduce the shared pool, a good benchmark is continually seeing 2-3MB of free memory.

On the other hand, if after an hour or so of beginning database operation, you see that virtually no free memory is left in the shared pool, or you are seeing ORA-4031 errors (that indicate definitions cannot find enough contiguous free space in the shared pool), increase the pool by 10% or more.

Troubleshooting

If you continuously see little or no free memory in the shared pool, you can do the following:

If a problem is found in Oracle8i or earlier:

- Edit the Init.ora file for the database.
- Increase the amount of shared_pool_size to a higher value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor the new value to see if performance improves.

If a problem is found in Oracle9i or later:

- Increase the size of the shared_pool_size parameter through use of the ALTER SYSTEM SET shared_pool_size command value. Take caution to not over-allocate; ensure enough free memory exists on server before increasing value.
- Monitor the new value to see if performance improves.
- If using an SPFILE, save the new configuration values so Oracle reuses them each time the database is stopped and re-started.

You can also use the ALTER SYSTEM FLUSHED SHARED_POOL command to remove all objects from the shared pool and start with a clean slate.

Session Leaders - Memory

Frequently one or two users can cause the majority of runtime problems that plague a database. The problem could be a runaway process, an untuned batch procedure, or other user-initiated operation. Frequently, user connections can get out of hand with memory consumption and extreme cases have caused headaches at both the database and operating system level (ORA-4030 errors).

NOTE: This statistic displays on both the Users performance category view and the Memory performance category view.

TIP: Click this statistic to drill down to the Leading Sessions tab of the Memory Detail view.

Metrics

If your database server does not have an overabundance of memory, you should periodically check to see two things:

1. Who are your heavy memory users
2. The total percentage of memory each user consumes.

If you see one or two users who have more than 5-15% of the total memory usage, you should further investigate the sessions to see what activities they are performing.

Troubleshooting

You can use the Session Leaders - Memory statistic to find the users with the greatest allocations of overall memory. Than you can drill down into the details to discover the memory components that make up each user's session. Pay particular attention to the amounts of memory usage associated with these areas:

Area	Definition
PGA	The Program Global Area is a private memory area devoted to housing the global variables and data structures for a single Oracle process.
UGA	The User Global Area contains session-specific information regarding open cursors, state information for packages, database link information, and more. Note that when using Oracle's Multi-threaded Server (MTS), the UGA can be moved up into the SGA.
Memory Sorts	Memory sorts contain a count of how many memory sorts a session has performed.

SGA

The SGA (System Global Area) is Oracle's memory structural area devoted to facilitating the transfer of data and information between clients and the Oracle database. The table below describes its main tunable components:

Component	Definition
Database Buffers	Maintains data blocks that are read from the database. Properly sizing the buffer cache goes a long way in improving response time performance.
Var Size	Holds object structure as well as code definitions. Setting the proper amount of memory in the variable size also assists a great deal in improving overall performance with respect to code execution and object references.
Redo Log Buffer	Buffers modifications that are made to the database before they are physically written to the redo log files.
Fixed Size	Contains a number of atomic variables, pointers, and other miscellaneous structures that reference areas of the SGA.

Metrics

None.

Memory Detail

The following tabbed pages are available in the Memory Detail view:

- Buffer Pool
- Data Dictionary
- Library Cache
- Leading Sessions
- Objects in Memory
- Sessions Overview
- Shared Pool

Data Dictionary Tab

- Metrics

Oracle's data dictionary is a component of the shared pool that contains system elements necessary for the parsing of SQL statements, security resolution, object definitions, and more.

The overall data dictionary cache hit ratio provides a key indicator in determining the performance of the shared pool, and shows how often object definitions are found in memory vs. having to read them in from disk. Because Oracle references the data dictionary many times when an SQL statement is processed, it is imperative that as much as possible of this vital reference information be kept in RAM.

The Data Dictionary tab shows the individual elements of the data dictionary along with their associated hit ratios. In versions 6.x of Oracle, these individual elements could be tuned, but in versions 7.x and later, the only main method for tuning them involves the adjustment of the entire shared pool setting. Although not tunable from an individual parameter level, each displayed element gives insight into which area of the data dictionary is either adding to or detracting from overall performance.

The table below describes the information available on this tab:

Column	Description
Parameter	The name of the individual data dictionary element.
Usage	The number of cache entries that contain valid data.
Gets	The number of requests for this element. To see what's happening to Dictionary Cache Gets in a given datasource, you can add this statistic to your Trends view.
Get Misses	The number of requests resulting in a cache miss To see what's happening to Dictionary Cache Misses in a given datasource, you can add this statistic to your Trends view.
Hit Ratio	The ratio of cache hits versus misses of total requests. The maximum is 100%.
Scans	The number of scan requests.
Scan Misses	The number of times that a scan failed to find the needed data in the cache.
Scan Completes	The number of times the list was scanned completely.
Modifications	The number of insert, update, and delete actions.
Flushes	The number of disk flushes.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

An overall high data dictionary cache hit ratio is desirable, as are high hit ratios in each individual parameter. You should strive for a hit ratio between 90-100%, with 95% being a good performance benchmark.

NOTE: When a database is first started, the data dictionary cache hit ratio is not at an optimal level because all references to object definitions are relatively new, and as such, must be placed into the shared pool. Look for hit ratios between 80-90% for new database startups. If, however, after an hour or two of steady database time, the data dictionary cache hit ratio and individual hit ratios, have not increased to desirable levels, you should look into the possibility of increasing the shared_pool_size parameter.

NOTE: Note that databases supporting applications that involve large number of objects (such as an Oracle Financials installation) should have larger than normal shared pools to support the required object definitions.

Buffer Pool Tab

- Metrics

Because data objects can be referenced with different usage patterns, Oracle8 and later offers the option to intelligently place objects into one of three buffer caches. The table below describes the types of buffer caches:

Cache	Description
KEEP	Designed to minimize misses in the buffer cache. Small objects that are frequently referenced are candidates for the KEEP buffer pool.
RECYCLE	Designed to avoid having large numbers of infrequently accessed blocks of data crowd out objects that need to be referenced in RAM, the RECYCLE pool is good for large objects that are scanned from beginning to end without the need for keeping all their data in RAM.
DEFAULT	The traditional cache for all other data objects.

NOTE: Unless you specify the KEEP or RECYCLE buffer cache, Oracle automatically places objects into the DEFAULT buffer cache.

The Buffer Pool Tab of the Memory Detail View calculates the hit ratios for each of the Oracle8 and later buffer caches so you can easily see how often the objects placed into the various caches are being referenced in memory. Examining how often data is satisfied from memory vs. disk will help you determine if the caches are large enough and if they are being used in an optimal manner. The table below describes the information available in this tab

Column	Description
Buffer pool name	The name of the Oracle buffer pool.
Buffer pool hit ratio	The overall hit ratio for the particular cache.

NOTE: The DEFAULT buffer cache will only be shown for those installations not using the specialized caches available in Oracle8 and later.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

The KEEP buffer pool should maintain a hit ratio as close to 100% as possible. However, the buffer pool hit ratio is not near 100% until the database has been up and running in a typical state for a short time.

A poor hit ratio for the RECYCLE buffer pool may not be a bad thing since there is little chance of reusing a block stored in the buffer pool before it is aged out.

NOTE: If you place objects into the KEEP buffer pool, you should periodically reexamine their object sizes to ensure that they are not growing to a physical state that will jeopardize the performance of the KEEP pool.

Library Cache Tab

- Metrics

The library cache holds commonly used SQL statements - basically database code objects. A method for improving performance in Oracle is to encourage the reuse of SQL statements so expensive parse operations may be avoided. The library cache assists this tuning effort.

The Library Cache tab provides insight into how efficiently the library cache is operating. The table below describes the information available in this section:

Column	Description
Namespace	The region of the library cache.
Gets	The number of times a lock was requested for objects in the particular namespace.
Get Hit Ratio	The percentage of times (with 100% being the maximum) that the object was found in the cache.
Pins	The number of times a pin was requested for objects of this namespace. To see what's happening to Library Cache Pins in a given datasource, you can add this statistic to your Trends view.
Pin Hit Ratio	The percentage of times (with 100% being the maximum) that pin requests were successful.
Reloads	The number of times a piece of the object had to be brought back in from disk to the cache, most likely because it was flushed from the shared pool. To see what's happening to Library Cache Reloads in a given datasource, you can add this statistic to your Trends view.
Invalidations	The number of times objects in this namespace were marked invalid because a dependent object was modified.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

NOTE: An overall high library cache hit ratio is a desirable thing. You should strive for a hit ratio between 95-100%, with 99% being a good performance benchmark for code reuse. When a database is first started, the library cache hit ratio, along with the individual region hit ratios, will not be at an optimal level because all code being used will be relatively new, and as such, must be parsed and placed into the shared pool. If, however, after a solid hour or two of steady database time, the library cache hit ratio has not increased to desirable levels, you should look into the possibility of increasing the shared_pool_size parameter.

To keep important code objects from being aged out of the library cache, you can use the DBMS_SHARED_POOL package to pin frequently used code objects in memory so they will always be there when needed.

Leading Sessions Tab

- Metrics

Frequently one or two users cause the majority of runtime problems that plague a database. The problem could be a runaway process, an untuned batch procedure, or other user-initiated operation. Sometimes, user connections can get out of hand with memory consumption, and extreme cases have caused headaches at both the database and operating system levels.

Performance Center displays information to help you find processes that are using the most memory on the server on the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view.

The table below describes the information available on the tabs for all supported database platforms:

Column	Description
User Name	The logon name the session is using.
SID	The unique Oracle identifier for the session.
Serial #	The serial number assigned to the session.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Machine	The name of the client machine name that the session is using.
PGA	The Program Global Area (PGA) is a private memory area devoted to housing the global variables and data structures for a single Oracle process, in KB.
UGA	The User Global Area (UGA) contains session specific information regarding open cursors, state information for packages, database link information, and more. When using Oracle's Multi-threaded Server (MTS), the UGA can be moved up into the SGA, in KB.
Memory Sorts	The number of memory sorts a session has performed.
Total Memory	The amount of memory (PGA + UGA) that the session is consuming, in KB.

NOTE: This information is available on both the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

If your database server does not have an overabundance of memory, you should periodically check two things:

1 Who are your heavy memory users

2 The total percentage of memory each user consumes

If you see one or two users who have more than 5-15% of the total memory usage, you should investigate the sessions further to see what activities they are performing.

Objects in Memory Tab

- Metrics

The Objects in Memory tab of the Memory Detail view shows objects currently in the library cache. This view lets you see exactly which objects are in the shared pool as well as their resource usage and activity levels.

The table below describes the information available on the Objects in Memory tab of the Memory Detail view:

Column	Description
Object Owner	The user account that owns the object.
Object Name	The name of the object.
Type	The type of object: INDEX, TABLE, CLUSTER, VIEW, SET, SYNONYM, SEQUENCE, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, TRIGGER, CLASS, OBJECT, USER, or DBLINK.
Sharable Memory	The amount of memory consumed by the object in the shared pool.
Loads	The number of times the object has been loaded into the cache. This count increases when an object has been invalidated.
Executions	The number of times that the object has been executed by a session thread.
Locks	The number of users actively locking the object.
Pins	The number of user actively pinning the object.
Kept	Whether or not the object has been pinned in memory with the DBMS_SHARED_POOL package.

NOTE: This information is available on both the Objects in Memory tab of the Memory Detail view and the Objects in Memory tab of the Objects Detail view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

If you see objects with large numbers of loads, this could signal that user sessions frequently use the objects causing the objects to prematurely age out of the cache. To fix this you could increase the size of the shared pool and/or pin the objects with the DBMS_SHARED_POOL package so they do not age out of the library cache.

Sessions Overview Tab

- Metrics

You can get more detail into dips in the overall buffer cache hit ratio by examining user I/O activity and users' individual cache hit ratios. Frequently you can pinpoint one or more accounts responsible for reducing performance and examine their SQL and other statistics to fix the situation.

The table below describes the information available on the Sessions Overview tab of the Memory Detail view:

Column	Description
Username	The logon name the session is using.
SID	The unique Oracle identifier for the session.
Serial #	The serial number assigned to the session.
Status	The status of the session, ACTIVE or INACTIVE.
Machine	The name of the client machine that the session is using.
O/S ID	The unique operating system identifier for the target session.
Logon Time	The timestamp when the session first logged on.
Blocked	Whether the session is blocked from doing work by another session.
Seconds Idle	The number of seconds since the last time the session actively performed work.
Hit Ratio	The percentage of times the session obtained data from memory vs. physical I/O activity. The maximum value is 100%.

NOTE: This information is available on both the Sessions Overview tab of the Memory Detail view and the Sessions Overview tab of the Users Detail view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Shared Pool Tab

What are the major components of the shared pool and how much memory does each take up? This question can be answered by viewing the Shared Pool section. You can easily see how much free memory is left in the pool as well as how much space all the other components consume. The table below describes the information available in this section:

Column	Description
Component	The major component of the SGA's shared pool.
Memory	The amount of memory the component is currently using, in bytes.
Percent of Shared Pool	The percentage of the shared pool the component uses.

I/O Statistics

The I/O home page includes the following sections:

- Archive Files Written Today
- Active Jobs
- Active Rollbacks
- Consistent Reads
- Index Scans
- Logical Changes

- Logical Reads
- Long Table Scans
- Physical Reads
- Physical Writes
- Redo Log Size
- Redo Wastage
- Redo Writes
- Rollback Gets
- Rollback Waits
- Session Leaders - I/O
- Short Table Scans
- User Commits
- User Rollbacks

Related Topics

I/O Detail

Home Page Statistics

Memory Page Statistics

Objects Page Statistics

OS Page Statistics

Space Page Statistics

Top SQL Page Statistics

Users Page Statistics

Network Statistics

Contention Statistics

Archive Files Written Today

- Metrics
- Troubleshooting

To allow for point-in-time recovery, Oracle writes copies of redo log information to disk. When a database is running in archive log mode, a DBA can (with proper backup techniques in place) recovery nicely from a database error and roll forward to almost any point in time needed, as long as the proper archive logs are in place.

Oracle's ARCH process handles the I/O needed to write these archive logs. The archive files written today statistic provides a count of how many archive files have been written for the current date.

> **TIP:** Click this statistic to drill down to the Archive view.

Metrics

Archive files written more than one per hour or half-hour could indicate a too small redo size (or above average data modification load).

Troubleshooting

If the redo log size is too small for the database, you can redefine to a larger size. You must take care when performing this job and consult the Oracle administrator documentation exact procedures on altering the redo log file size.

User Object Accesses

This section of the I/O performance category view details some of the most important statistics relating to object accesses. The table below describes the statistics:

Statistic	Description
Long Table Scans	The number of scans of tables consisting of five or more data blocks. This statistic displays on both the I/O performance category view and the Objects performance category view. For more information on long table scan rows, see Long Table Scan Rows.
Short Table Scans	The number of scans of tables consisting of fewer than five data blocks.
Index Scans	The number of object access by a table's ROWID - usually accomplished through an index. For more information on index scans, see Index (ROWID) Accesses.
User Commits	A user commit, which normally signaling the end of a transaction, causes the LGWR process to write modified blocks to disk.
User Rollbacks	The number of times the user transaction manually issued the times the ROLLBACK SQL statement. Click this statistic to drill down to the Rollback Activity tab of the I/O Detail view.
Active Jobs	The number of active jobs, which are scheduled through the use of the Oracle job queue, indicate how many jobs are currently running in the queue. Click this statistic to drill down to the Active Jobs tab of the I/O Detail view.

Metrics

Depending on the statistics found, here are some general user I/O guidelines:

- Avoid unnecessary scans on large tables. Also, high counts of large table scans are a signal to you as a DBA to both investigate the use of more indexes and to review SQL access through EXPLAIN plans.
- Small table scans are actually a good thing because Oracle can often cache the entire table in a single I/O operation.
- Large numbers of index scans are normally desirable too, because they typically indicate the fastest possible resolution to data access requests.
- Keeping an eye on user commits helps give you one of the truest transaction rate indicators in your database.
- Large numbers of user rollbacks can be undesirable because they indicate that user transactions are not completing for one reason or another.
- An increasing number of active jobs can indicate a backup of job activity in the Oracle job queue.

Troubleshooting

Listed below are some methods you can use to avoid unnecessary large table scans:

- Try not to use SQL statements that include the NOT IN, NOT LIKE, <>, IS NULL operators because they typically suppress the use of indexes.
- When referencing concatenated indexes with queries, be sure that you use the leading column in the index. If you do not, the index will not be used at all.
- Avoid using functions in WHERE predicates. If you must use functions, and you are using Oracle8i, investigate the use of function-based indexes.

If you must use table scans, try and make use of the following Oracle features to increase scan performance as much as possible:

- Parallel query
- Partitioning
- Table CACHE parameter
- KEEP buffer pool (Oracle8)

Active Rollbacks

- Metrics

Oracle writes data to individual rollback segments to undo changes made to the Oracle database from within a transaction. You can also use these to maintain read consistency for multiple users of modified data. Because Rollback Segments read and write data, they can become very hot areas for I/O. This statistic is a count of how many rollback segments are actively involved in transactions for the database.

NOTE: This statistic displays on both the I/O performance category view and the Objects performance category view.

TIP: Click this statistic to drill down to the Rollback Activity tab of the I/O Detail view.

Metrics

When you drill down into this statistic and see that most or all rollback segments are involved in transactions, you should consider adding more rollback segments to the system.

Session Leaders - I/O

- Metrics

Heavy I/O activity in a system sometimes indicates that the user connections are contributing somewhat equally to the overall load. More often than not, however, one or two user connections are responsible for 75% or more of the I/O activity. It could be that your system is running a large batch load or other typical processes, which are perfectly okay. On the other hand, it could be a runaway process or other rogue connection, which you need to track down and possibly eliminate.

The leading I/O session's display lets you see who the leading sessions are in your system with respect to I/O.

NOTE: This statistic displays on both the Users performance category view and the I/O performance category view.

TIP: Click this statistic to drill down to the Leading Sessions tab of the I/O Detail view.

Metrics

One or two users consuming more than 75% of the total I/O load could indicate a runaway or improper process. You can drill into the I/O activity of all users and quickly see if this is the case.

Redo Wastage and Redo Log Size

Oracle redo logs are hotbeds of I/O activity in databases that store heavily modified data. Oracle uses redo log files to perform database recovery in the event of a system crash and it writes redo logs in a cyclical fashion: each log file is filled up before Oracle moves on to the next file. The Redo Log Size statistic reflects the total amount of redo generated in bytes since the last refresh. Redo Wastage indicates the number of bytes that were wasted by redo log buffers being written before they were full.

> **TIP:** Double-click these statistics to drill down to the LGWR/DBWR Detail tab of the I/O Detail view.

Metrics

A high percentage of wasted bytes to overall redo bytes can help you identify if redo wastage is a problem on your system.

Troubleshooting

Sometimes heavy redo wastage occurs when the log_checkpoint_interval parameter is set too high. If you think this is the case for your system, then:

- Edit the Init.ora file for the database.
- Increase the amount of log_checkpoint_interval to a lower value.
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor new value to see if performance improves.

Rollback Gets and Rollback Waits

Oracle writes data to individual rollback segments to undo changes made to the Oracle database from a transaction. You can also use these segments to maintain read consistency of data modified by multiple users. Because rollback segments read and write data, they can become very hot areas for I/O.

A count of rollback gets defines the number of times a process went to a rollback header to glean information. The count of rollback writes is the number of bytes written to all rollback segments.

> **TIP:** Double-click these statistics to drill down to the Rollback Activity tab of the I/O Detail view.

Metrics

Taking the number of rollback writes and dividing them by the number of rollback gets and waits yields the overall rollback contention ratio, which indicates how bad contention is for rollback segments. A count greater than 1% typically means that you should add more rollback segments to the database.

Physical I/O

Physical I/O consists of Oracle going to disk to gather or write data. The database writer (DBWR) and log writer (LGWR) processes typically perform all the I/O work in the database. You can also use other processes like the checkpoint and archive processes (CKPT and ARCH). Performance Center shows three key indicators of physical I/O. The table below describes these indicators:

Indicator	Description
Physical Reads	Physical Reads is the total number of physical reads performed on all datafiles since the last refresh.
Physical Writes	Physical Writes is the total number of times the DBWR process has performed writes to various database datafiles since the last refresh
Redo Writes	Oracle's redo logs are hotbeds of I/O activity in databases that store heavily modified data. Redo log files are used to perform database recovery in the event of a system crash. Redo logs are written to in a cyclical fashion - each log file is filled up before Oracle moves on to the next file. The redo writes statistic reflects the total number of redo writes by the LGWR process since the last refresh

NOTE: For more information on physical I/O statistics, see Logical Vital Signs.

TIP: Click this statistic to drill down to the I/O by Tablespace tab of the I/O Detail view.

Metrics

The table below describes metrics for Physical I/O statistics:

Statistic	Metrics
Physical Reads	Large numbers of physical reads could reflect a too small data/buffer cache. The buffer cache hit ratio is a better indicator of overall logical vs. physical I/O.
Physical Writes	Wait events related to I/O activity are good indicators of physical I/O problems. These events include `db file parallel write` and `db file single write`.

Troubleshooting

Doing the following can negate large numbers of continuous physical reads:

- Increasing the size of the data/buffer cache.
- Pinning often-referenced objects in memory by using the KEEP buffer pool (Oracle 8 and higher.)
- Placing heavily scanned objects in larger blocksize tablespaces (16-32KB). For Oracle9i or later.
- Tune SQL statements for better efficiency.

Logical I/O

Logical I/O refers to data access performed in memory. The database writer (DBWR) and log writer (LGWR) processes typically perform all the I/O work in the database. You can also use other processes like the checkpoint and archive processes (CKPT and ARCH). Performance Center shows three key indicators of logical I/O. The table below describes these indicators:

Indicator	Description
Logical Reads	Logical Reads is the total number of db block gets and consistent gets (data read from memory) since the last refresh.
Logical Changes	Logical Changes is the total number of changes that were made to all blocks in the SGA that were part of an update or delete operation. These changes generate redo log entries and are permanent if the transaction is committed. The number of logical changes is an approximate indication of total database work.
Consistent Reads	Consistent Reads is the total number of times a consistent read was requested for a database block. Such a read is performed from Oracle's rollback segments.

NOTE: For more information on logical I/O statistics, see Logical Vital Signs.

TIP: Click this statistic to drill down to the I/O by Tablespace tab of the I/O Detail view.

Metrics

Regarding raw logical I/O counts, no hard-core metrics exist. However, because physical I/O takes longer to complete than logical (memory) I/O, you should minimize physical read operations when possible. The buffer cache hit ratio is a better indicator of overall logical vs. physical I/O.

Troubleshooting

While logical I/O is still up to 1,400 times faster than physical disk access, it would be wise to investigate the top logical I/O process using Performance Center and see what SQL it is executing. If one process on the system is consuming between 25-50% of the overall amount, their SQL might require tuning.

I/O Detail

The following tabbed pages are available on the I/O Detail page:

- I/O by Datafile
- Log I/O - DBWR/LGWR
- Rollback Activity
- I/O by Tablespace
- Leading Sessions
- Active Jobs Tab

Active Jobs Tab

- Metrics

Scheduled using the Oracle job queue, custom batch jobs can be set up to run whenever a DBA would like. The Active Jobs tab of the I/O Detail view presents the active jobs that are running and indicates the number of jobs currently running in the queue.

The table below describes the information available on the Active Jobs tab of the I/O Detail view:

Column	Description
SID	The identifier of the session executing the job.
Job ID	The identifier of the job.
Start Time	The time that the job was started.
Submitted By	The user account that submitted the job.
Run As	The user account whose privileges run the job.
Parse As	The user account that parsed the job.
Next Run	The next scheduled job run time.
Job Contents	The body of the anonymous PL/SQL block that this job executes.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Viewing jobs with long run times could indicate a job caught in a loop or experiencing a problem.

Leading Sessions Tab

- Metrics

When a system undergoes heavy I/O activity, sometimes you find that all the user connections are contributing somewhat equally to the overall load. Frequently, however, one or two user connections are responsible for 75% or more of the I/O activity. It could be that a large batch load or other typical process is running that is perfectly okay for your system. Alternatively, it could be a runaway process or other rogue connection that you should track down and possibly eliminate.

Performance Center displays information to help you find processes that are using the most memory on the server on the Leading Sessions tab of the I/O Detail view and the I/O tab of the Top Sessions view.

The table below describes the information available on the tabs for all supported database platforms:

Column	Description
Username	The logon name the session is using.
SID	The unique Oracle identifier for the session.
Serial #	The serial number assigned to the session.
Status	The status of the session, ACTIVE or INACTIVE.
Machine	The name of the client machine name that the session is using.
Reads	The number of physical reads.
Writes	The number of physical writes.
Total I/O	The total of all physical I/O operations for the session.

NOTE: This information is available on both the Leading Sessions tab of the I/O Detail view and the I/O tab of the Top Sessions view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Pinpointing sessions with abnormally high I/O activity relative to other sessions in the system could aid in ferreting out accounts dragging down overall system performance. You should examine the activity of each session to determine the system workload and to determine if you can reduce the workload or tune the system for better performance.

I/O by Tablespace Tab

- Metrics

Physical I/O consists of Oracle going to disk to gather or write data. Logical I/O refers to data access performed in memory. The database writer (DBWR) and log writer (LGWR) processes typically perform all the I/O work in the database. Other processes like the checkpoint and archive processes (CKPT and ARCH) may also be used.

The Tablespace I/O tab displays details concerning the physical I/O activity at the tablespace level. The table below lists the information available on this tab:

Column	Description
Tablespace	The name of the tablespace.
Physical Reads	The cumulative number of physical reads.
Physical Writes	The cumulative number of physical writes.
Block Reads	The cumulative number of physical block reads.
Blocks Written	The cumulative number of physical block writes.
Read Time	The time spent reading from the tablespace (in hundredths of seconds).
Write Time	The time spent writing to the tablespace (in hundredths of seconds).

Metrics

Generally, you want to see much more logical I/O activity than physical I/O, at least with respect to reads, although this in and of itself is a guarantee of good I/O performance. Seeing logical and physical reads keeping pace with one another is a sure sign that the Oracle SGA's buffer cache is sized too small or that needless, large table scans are occurring, which cause blocks of data to be continually read in and flushed out of the buffer cache.

Other telltale signs of trouble brewing are large amounts of activity visible in user's TEMP tablespaces. The normal interpretation of such a thing is that a large number of disk sorts are taking place (perhaps because the Init.ora/spfile parameter SORT_AREA_SIZE may be set too small).

For more information, see I/O Home Page.

I/O by DatafileTab

- Metrics

Physical I/O consists of Oracle going to disk to gather or write data. Logical I/O refers to data access performed in memory. The database writer (DBWR) and log writer (LGWR) processes typically perform all the I/O work in the database against the physical datafile used to hold information The checkpoint and archive processes (CKPT and ARCH), also perform I/O work in the database.

The Datafile I/O tab displays details concerning the physical I/O activity at the datafile level. The table below lists the information available on this tab:

Column	Description
Datafile	The name of the datafile.

Column	Description
Datafile Status	The availability of the datafile.
Tablespace	The name of the tablespace.
Physical Reads	The cumulative number of physical reads.
Physical Writes	The cumulative number of physical writes.
Block Reads	The cumulative number of physical block reads.
Blocks Written	The cumulative number of physical block writes.
Read Time	The time spent reading from the tablespace (in hundredths of seconds).
Write Time	The time spent writing to the tablespace (in hundredths of seconds).

Metrics

Generally, you will want to see much more logical I/O activity than physical I/O, at least with respect to reads, although this in and of itself is a guarantee of good I/O performance. Seeing logical and physical reads keeping pace with one another is a sure sign that the Oracle SGA's buffer cache is sized too small or that needless, large table scans are occurring, which cause blocks of data to be continually read in and flushed out of the buffer cache.

Other telltale signs of trouble brewing are large amounts of activity visible in user's TEMP tablespaces. The normal interpretation of such a thing is that a large number of disk sorts are taking place (perhaps because the Init.ora/spfile parameter SORT_AREA_SIZE may be set too small). One last thing to watch is high amounts of activity on datafiles housed on the same directories or file systems. This usually indicates a "hot" drive and could represent disk contention. If you see one or more hot drives, you should consider relocating one or more datafiles to lesser-used physical devices.

For more information, see I/O Home Page.

Rollback Activity Tab

The Rollback I/O tab includes the following sections:

- Rollback Activity
- Rollback/Optimal Detail

For more information, see I/O Home Page.

Rollback Activity Grid

- Metrics

To undo changes made to the Oracle database from within a transaction, Oracle writes data to individual rollback segments. Oracle also uses these segments to maintain read consistency for multiple users of data that is being modified. Because Oracle reads from and writes data to rollback segments, they can become very hot areas for I/O.

The Rollback Activity Detail grid presents everything necessary to view and troubleshoot rollback problems. The first grid displays rollback activity details and the second grid displays rollback/optimal details. The table below describes the information available in the Rollback Activity Detail grid on the Rollback Activity tab of the I/O Detail view:

Column	Description
Name	The name of the rollback segment.
Size	The size of the rollback segment in KBs.
Shrinks	The number of times the rollback segment has decreased in size.

Column	Description
Extends	The number of times the rollback segment has increased in size.
Gets	The number of header gets (the segment has been used).
Waits	The number of header waits.
Writes	The number of bytes written to the rollback segment.
Active	Indicates whether the rollback segment is active (non zero) or not (zero value).
Status	Indicates the status of the rollback segment, with the two main results being OFFLINE (a segment is offline and unavailable for transactions) and ONLINE (a rollback segment is online and available for transactional use).
High Water Mark	The largest size that the rollback segment has ever grown to.

Rollback/Optimal Detail Grid

- Metrics

The lower grid presents data on each rollback segment's current status with respect to its optimal size setting (the DBA-imposed size that the segment "should" be).

The table below describes the information available in the Rollback/Optimal Detail grid on the Rollback Activity tab of the I/O Detail view:

Column	Description
Rollback Name	The name of the rollback segment.
Size	The size of the rollback, in bytes.
Optimal Size	The optimal setting for the rollback segment. NULL if the DBA has not imposed an optimal setting on the rollback segment.
High-Water Mark	The largest size to which the rollback segment has ever grown, in bytes.
Pct Above Optimal	The percentage above the optimal setting to which the rollback segment ever grown.

NOTE: If a rollback segment has no OPTIMAL setting, it has NULL values for most values in the Rollback/Optimal Detail grid.

TIP: To configure a grid to display row numbers, use the Options Editor.

Metrics

To properly tune rollback I/O, you must first make sure that you have enough segments to accommodate the workload of the database. Constantly seeing a count of active rollback segments equal to or near the number of rollbacks defined for the database is an indicator that you should create more. An overall rollback contention ratio of 1% or greater is an indicator of too few rollbacks. Seeing wait counts greater than zero for each rollback segment is further evidence that you should create more rollback segments. Oracle9i provides the UNDO tablespace to automatically generate and eliminate the ‘correct’ number of rollback segments for a system given a certain workload.

After ensuring that enough rollback segments exist in the database, you should then turn your attention to the question of sizing. Dynamic rollback extension can take a toll on performance when rollback segments are consistently enlarged to accommodate heavy transaction loads. Seeing rollback segments undergoing numerous extends and shrinks (as Oracle returns a segment back to its OPTIMAL setting), as well as rollback segments having current or high-water mark sizes greater than their OPTIMAL setting usually is a good indicator that they should be permanently enlarged. Again, Oracle9i’s automatic undo management can assist in this process.

DBWR/LGWR Tab

The DBWR/LGWR tab includes Database Writer (DBWR/LGWR) detail and Redo Log detail.

For more information, see I/O Home Page.

Database Writer Detail

- Metrics

The database writer process (DBWR) handles the flow of information from Oracle's physical datafiles to and from the various memory structures in the system global area (SGA). On platforms that support it, you can configure and use multiple DBWR processes. The log writer (LGWR) process manages the information contained in Oracle's online redo log files and redo log buffer area. On the LGWR/DBWR tab of the I/O Detail view you can view statistics relating specifically to both of these critical Oracle background processes and see if they are running efficiently. The table below lists the information available in this section:

Column	Description
Statistic	The specific metric for the database writer.
Value	The value for the statistic.

Metrics

Of all the statistics presented for the DBWR process, the summed dirty queue length statistic deserves attention. Non-zero values typically indicate buffers left in the write queue after a write request and may indicate that the DBWR process is falling behind. For the LGWR process, nonzero values for the redo log space requests and redo log space wait time statistics could be a cause for concern. Redo log space requests reflect the number of times a user process waited for space in the redo log buffer, while the redo log space wait time is the total wait time in milliseconds. Both could indicate the presence of contention in the redo log buffer. Possible remedies include increasing the LOG_BUFFER size in the SGA.

Redo Wastage

Oracle's redo logs are hotbeds of I/O activity in databases that store heavily modified data. Redo log files are used to perform database recovery in the event of a system crash. Redo logs are written to in a cyclical fashion - each log file is filled up before Oracle moves on to the next file. The Redo Wastage section shows how many bytes were wasted by redo log buffers being written before they were full.

Metrics

Viewing a high percentage of wasted bytes to overall redo bytes can help you identify if redo wastage is a problem on your system.

Steps

Sometimes heavy redo wastage occurs when the log_checkpoint_interval parameter is set too high. If you think this is the case for your system, then

1. Edit the Init.ora file for the database.
2. Change the amount of log_checkpoint_interval to a lower value.
3. Cycle the Oracle server when possible to allow the new value to take effect.
4. Monitor new value to see if performance improves.

Space Statistics

The Space performance category view displays the following vital Oracle space statistics:

- Fragmentation Leaders
- Schema Leaders - Space
- Tablespace Overview

Related Topics

Space Detail

Home View Statistics

I/O Page Statistics

Memory Page Statistics

Objects Page Statistics

Users Page Statistics

Network Statistics

Contention Statistics

Tablespace Overview

- Metrics
- Troubleshooting

The Oracle tablespace is the logical container for the physical storage needs of a database. The tablespace overview displays details for all the tablespaces for a particular database, including their total, used, free space, and percent free, as well as if each can automatically extend to support incoming space requests.

> **TIP:** Click this statistic to drill down to the Space Usage tab of the Space Detail view.

Metrics

If any tablespace's free space percent amount goes below 10%, and at least one tablespace's datafiles does not have AUTOEXTEND enabled (or the datafile has reached its extend limit), you should take action to ensure that the tablespace does not run out of available free space.

Troubleshooting

There are two things you can do to ensure that a tablespace does not run out of available free space:

1. First, you should look into the use of Oracle's AUTOEXTEND feature. AUTOEXTEND lets you give an Oracle tablespace the ability to auto-grow when it has exhausted the free space contained within. You can let a tablespace grow in an unlimited fashion or put constraints on it to stop at a certain point. You can also dictate how much more free space the tablespace gets each time it needs more space than is available. However, AUTOEXTEND enabled for a tablespace does not mean that you cannot run out of space. Remember you still have the physical server limitations to contend with. Make sure you (or your sysadmin) keep a careful eye on the server drives that house your Oracle database files for available free space.
2. If the free space on a server drive nears its limit, disable AUTOEXTEND for the datafile(s) that are on the target drive, and use the ALTER TABLESPACE … ADD DATAFILE command to place a new datafile for the tablespace on another drive that has more free space to offer.

TIP: AUTOEXTEND is not a replacement for proactive space planning. When extra space is needed by the database, and AUTOEXTEND is activated by Oracle, performance slows as Oracle allocates more space for the tablespace. Avoiding AUTOEXTEND aids performance, albeit in a small way.

Fragmentation Leaders

The Fragmentation Leaders table shows the tablespaces datafiles suffering from the highest levels of free space fragmentation in the database. Tablespaces are made up of object segments and space extents. Extents are either allocated to object segments or are free. When a tablespace is initially populated, all objects are neatly packed together in the front of the tablespace and all remaining free space is in one free chunk at the end. As objects grow (or extend) they are given new extents of space in the tablespace. If objects are dropped, pockets of free space begin to appear throughout the tablespace. These pockets of space take one of two forms. The table below describes these forms:

Free Space	Description
Honeycombs	Pockets of free space that are adjacent to one another.
Bubbles	Pockets of free space that are trapped between object extents in the tablespace.

TIP: Click this statistic to drill down to the Fragmentation tab of the Space Detail view.

NOTE: This information is available on both the Fragmentation Leaders table of the Space performance category view and the Fragmentation tab of the Space Detail view.

Metrics

If you see a tablespace that has many chunks of free space, determine if the tablespace is experiencing honeycomb or bubble fragmentation. You can handle honeycomb fragmentation quite easily, whereas bubble fragmentation is more difficult to solve.

Troubleshooting

You can eliminate honeycomb fragmentation with the ALTER TABLESPACE…COALESCE command. Issuing this command combines all pockets of adjacent free space into one extent. Each database maintenance plan should include a job that coalesces all the free honeycombs in a tablespace into one free chunk. Although Oracle is supposed to perform this operation automatically through the SMON process, it requires you to have the PCTINCREASE parameter of the tablespace set to a nonzero value. Having PCTINCREASE set to a value greater than zero encourages tablespace fragmentation through disparately sized extents. Plus, using SMON in this way is not efficient or entirely reliable.

You can temporarily solve bubble fragmentation by performing a total tablespace reorganization. A better long-term solution for Oracle databases 8.1.5 or later is to convert tablespaces over to locally managed tablespaces. With locally managed tablespaces you either specify the initial extent size and let Oracle automatically size all other extents, or specify a uniform extent size for everything. Problems caused by fragmentation then become a thing of the past.

To help stave off fragmentation problems:

- Set PCTINCREASE to zero for all tablespaces and objects to promote same-sized extents.
- Specify equal-sized allotments for your INITIAL and NEXT object storage parameters.
- Group objects with like growth and storage needs together in their own tablespaces.

You should also avoid fragmentation in the SYSTEM tablespaces. The best ways to do this include:

- Ensure no user has a DEFAULT or TEMPORARY tablespace assignment of SYSTEM.
- Ensure no user has a quota set for SYSTEM.
- Ensure no user has been granted the UNLIMITED TABLESPACE privilege.

Schema Leaders - Space

The Space Leaders chart details the schemas in the database that own the most space. In databases where object owners can be many, it is frequently a good idea to take a quick look at which schemas are leading the way in terms of space usage. This is not restricted to production environments only, but can also be extended to dynamic development systems. This is useful when you have many developers in a database that have their own sets of objects. Developers sometimes have a bad habit of creating objects (or copies of objects) that they leave in a database even though they are not using them.

> **TIP:** Click this statistic to drill down to the Space Usage tab of the Space Detail view.

Metrics

None.

Troubleshooting

If you see a user who owns many objects, or copies of objects, reclaim space in your database by contacting the user to see if those objects can be dropped.

Space Detail

The following tabbed pages are available on the Space Detail page:

- Fragmentation Tablespace Growth
- Extents Deficits

- Object Summary
- Space Usage
- Tablespace Map

Tablespace Map Tab

- Metrics

Tablespaces comprise object segments and space extents. Extents are either free inside a tablespace or allocated to object segments. When a tablespace is initially populated, Oracle neatly packs all objects together in the front of the tablespace and all remaining free space is in one free “chunk” at the end. Unfortunately, this is not how things continue to be in a tablespace. As objects grow, or extend, the database gives them new extents of space in the tablespace. If you drop objects, pockets of free space begin to appear throughout the tablespace. These pockets of space are either honeycombs or bubbles.

The Tablespace Map tab of the Space Detail view lets you see all the extents each object in a tablespace takes up and all the free extents. The map is in block id order, letting you view the tablespace from beginning to end. The table below describes the information available on the Tablespace Map tab of the Space Detail view:

Column	Description
Owner	The owner of the object. NULL if free space occupies the extent.
Object Type	Type of object. NULL if free space occupies the extent.
Object Name	The name of the object occupying the extent. NULL if free space occupies the extent.
Block ID	The starting block number of the extent.
Bytes	The amount of space consumed by the extent, in bytes.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

The tablespace map identifies honeycomb problems with two adjacent chunks of free space. You can eliminate honeycomb fragmentation with the ALTER TABLESPACE…COALESCE command. This command combines all pockets of adjacent free space into one extent. This is important because when an object is newly introduced to a tablespace (or an existing object needs to extend), and a contiguous chunk of free space does not exist to accommodate an object's INITIAL or NEXT size allocation, Oracle must manually coalesce all available honeycombs to try and form a large enough free chunk. This is a performance hit. If possible, you should try to minimize performance hits.

Bubble fragmentation is a more serious matter and is normally only corrected through tablespace or database reorganization. The standard technique is to perform an export of all objects in the tablespace, drop the tablespace, and then import all the objects back into the tablespace. However, this technique may just treat the symptom and not the cause of bubble fragmentation. The real issue is to address the reuse of space extents within a tablespace so that bubble fragmentation does not occur in the first place.

Oracle8 offers the concept of locally managed tablespaces, which can all but eliminate tablespace fragmentation. It totally does away with the storage parameters of MINEXTENTS, MAXEXENTS, PCTINCREASE, and NEXT. With locally managed tablespaces you either specify the initial extent size and let Oracle automatically size all other extents, or specify a uniform extent size for everything. Problems caused by fragmentation then become outdated.

A few suggestions to manually to help stave off fragmentation problems include:

- Set PCTINCREASE to zero for all tablespaces and objects to promote same-size extents.
- Specify equal-size allotments for your INITIAL and NEXT object storage parameters.

- Group objects with like growth and storage needs together in their own tablespaces.

The SYSTEM tablespace is the major hotbed tablespace for Oracle activities. Therefore, more than any other tablespace, you should avoid fragmentation problems in your SYSTEM tablespace. The easiest way to avoid this is to not allow any user (even the default DBA ID's SYS and SYSTEM) to have access to it. There are three ways to accomplish this:

1 Ensure no user has a DEFAULT or TEMPORARY tablespace assignment of SYSTEM.

2 Ensure no user has a quota set for SYSTEM.

3 Ensure no user has been granted the UNLIMITED TABLESPACE privilege.

Object Summary Tab

- Metrics

What takes up space in a database? Simply, objects. In an Oracle database, the primary users of space are tables, indexes, and rollback segments. Under certain circumstances, other objects like materialized views and snapshots can come into play. With respect to space, as a DBA you should keep track of your object's growth and space characteristics. As with tablespaces, you want to keep an eye on fragmentation as well, unless locally managed tablespaces are being used (Oracle 8.1.5 or later).

The Object Summary tab of the Space Detail view displays object sizing information for selected tablespaces. The table below describes the information available on the Object Summary tab of the Space Detail view:

Column	Description
Owner	The user account who owns the object.
Object	The object name.
Object type	The type of object.
Kilobytes	The size of the object, in KB.
Extents	The number of extents used by the object.
Percent of tablespace used	The space in the tablespace consumed by the object, as a percentage.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

To avoid object growth errors in your database, you should consider setting the MAXEXTENTS parameter for your objects (which acts as a growth limit for the object) to a very high value or UNLIMITED if allowed by your version of Oracle. This lets you allocate as many extents as the objects need for growth. It is a good practice to monitor the amount of space your objects use and the number of extents they possess. This clues you in to what the dynamic objects in your database are.

One thing you can face with respect to allowing this type of unchecked extent growth is that your objects can reach a point of needing a new extent of space, but not enough free space exists in its parent tablespace to accommodate the enlargement request. It is good practice to periodically sweep your database for objects that can encounter this error on their next extension.

Understand that allowing unrestricted extent growth can save you from extent allocation errors, but introduces object fragmentation. What can you do if you have objects with high numbers of extents and are not using locally managed tablespaces? For tables and indexes, you have three options:

1. Use Oracle's export/import utility to export, drop, and then re-import the fragmented objects back into the database with the export parameter COMPRESS=Y. This brings the object back into the database with one large extent. Take care that large enough free chunks of available free space exist to accept the objects back in, or you can experience allocation errors.
2. With Oracle8, you can use the ALTER TABLE…MOVE command to reorganize a table back into one extent in a very safe and efficient manner.
3. Use ALTER INDEX…REBUILD to reorganize indexes that have moved into multiple extents.

For rollback segments, you can specify an OPTIMAL setting for each rollback segment. Setting OPTIMAL for the rollback gives the segment the opportunity to shrink back to whatever extent limit you would like to keep the rollback segment. The database periodically does this, but you can manually issue the ALTER ROLLBACK…SHRINK command.

Tablespace Growth Tab

- Metrics

Growing tablespaces do not have to spell problems for a DBA. Of course, up front planning is the key to sizing tablespaces correctly. Unfortunately, DBAs may not have all the information they need at tablespace creation time (or they may have the wrong information), so a tablespace can approach the end of its free space from time to time. The DBA can allow a tablespace to grow automatically (AUTOEXTEND) to prevent an out-of-space condition. Enabling AUTOEXTEND for a tablespace is quite reassuring for a DBA, but it introduces a new concept for the tablespace: monitoring data file growth. You should monitor your tablespaces that have AUTOEXTEND set so you can get an idea of the growth that is occurring in your database. Monitoring them lets you perform some mini-capacity planning and helps you get a jump-start on server upgrades.

The Tablespace Growth tab displays growth statistics for the database currently being monitored. The table below lists the information available on this tab:

Column	Description
Tablespace Name	The name of the tablespace.
Datafile Name	Thee name of the tablespace's datafile.
Auto Extend?	Indicates whether the tablespace has one or more datafiles that have the AUTOEXTEND feature set (allowing the datafile to automatically grow).
Original Size	The starting size for the physical datafile.
Current Size	The current size of the physical datafile.
Growth	The percentage of growth experienced for the datafile (if any).

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Using AUTOEXTEND can be especially important for tablespaces devoted to disk sorting. Large data warehouses often must endure large disk sort operations. Having AUTOEXTEND enabled for tablespaces used for temporary segments (sort activity) helps large sort queries to complete when they might otherwise fail due to running out of sort space.

Just having AUTOEXTEND enabled for a tablespace does not mean you cannot run out of space. Remember, you still have to contend with the physical server limitations. Make sure you (or your sysadmin) keep a careful eye on the server drives that house your Oracle database files for available free space. If the free space on a server drive nears its limit, disable AUTOEXTEND for the datafile(s) that are on that drive, and use the traditional ALTER TABLESPACE…ADD DATAFILE command to place a new datafile for the tablespace on another drive that has more free space to offer.

Fragmentation Tab

- Metrics
- Troubleshooting

Tablespaces are made up of object segments and space extents. Extents are either allocated to object segments or are free. When a tablespace is initially populated, all objects are neatly packed together in the front of the tablespace and all remaining free space are in one free chunk at the end. Unfortunately, this is not how things continue to be in a tablespace. As objects grow (or extend) they are given new extents of space in the tablespace. If you drop objects, pockets of free space begins to appear throughout the tablespace. These pockets of space are either honeycombs or bubbles. Honeycombs are not so difficult to deal with, but bubbles are another story.

The Fragmentation tab displays two different views of tablespace fragmentation. The first grid displays fragmentation at the tablespace level and the second grid displays fragmentation at the datafile level. The table below describes the information available in the Tablespace Fragmentation grid:

Column	Description
Tablespace Name	The tablespace name.
Free Space (MB)	The total amount of free space in MB for the tablespace.
Free Chunks	The total number of free chunks in the tablespace.
Largest Chunk (MB)	The largest free chunk (in MB) for the tablespace.

NOTE: Performance Center displays the fragmentation grid in two places, the Space performance category view and the Space Detail view.

The table below describes the information available in the Datafile Fragmentation grid on the Fragmentation tab:

Column	Description
Tablespace	The tablespace name.
Datafile	The name of the datafile.
Datafile Status	The status (AVAILABLE or INVALID) of the individual data file. A status of INVALID means that the file number is not in use. This could happen to a file that was part of a tablespace that was dropped.
Free Space	The amount of free space for the datafile, in MB.
Free Chunks	The number of free chunks in the datafile.
Largest Chunk	The largest free chunk for the datafile, in MB.

Column	Description
Datafile	The name of the datafile.
Autoextend	Indicates whether the datafile can automatically grow in size.
Tablespace	The tablespace name.
Free Chunks	The number of free chunks in the datafile.
Largest Chunk	The largest free chunk (in MB) for the datafile.

NOTE: This information is available on both the Fragmentation Leaders table of the Space performance category view and the Fragmentation tab of the Space Detail view.

TIP: To configure a grid to display row numbers, use the Options Editor.

Metrics

To spot and correct fragmentation in your tablespaces, you should periodically monitor the fragmentation levels of your tablespaces at a global level. Doing so helps you quickly spot tablespaces that are experiencing fragmentation issues. Seeing a tablespace with only one chunk of free space is a sign that a tablespace is not having fragmentation problems. Seeing a tablespace with a couple of free chunks may not be a big deal either, because the tablespace could be made up of more than one datafile. Each datafile has its own chunk or chunks of free space.

If you see a tablespace that has many chunks of free space, the next thing to do is drill down into it and find out if the tablespace is experiencing honeycomb or bubble fragmentation. Honeycomb fragmentation occurs when pockets of free space exist that are adjacent to one another. Bubbles are pockets of free space that are trapped between object segment extents.

You can eliminate honeycomb fragmentation with the ALTER TABLESPACE...COALESCE command. This command combines all pockets of adjacent free space into one extent. It is important to do this because when an object is newly introduced to a tablespace (or an existing object needs to extend), and a contiguous chunk of free space does not exist to accommodate an object's INITIAL or NEXT size allocation, Oracle must manually coalesce all available honeycombs to try and form a large enough free chunk. This is a performance hit. If possible, you should try to minimize performance hits.

Bubble fragmentation is a more serious matter and is normally only corrected through tablespace or database reorganization. The standard technique is to perform an export of all objects in the tablespace, drop the tablespace, and then import all the objects back into the tablespace. However, this technique may just treat the symptom and not the cause of bubble fragmentation. The real issue is to address the reuse of space extents within a tablespace so that bubble fragmentation does not occur in the first place.

Oracle8 and later offers the concept of locally-managed tablespaces, which can all but eliminate tablespace fragmentation. It totally does away with the storage parameters of MINEXTENTS, MAXEXTENTS, PCTINCREASE, and NEXT. With locally managed tablespaces you either specify the initial extent size and let Oracle automatically size all other extents, or specify a uniform extent size for everything. Problems caused by fragmentation then become a thing of the past.

Troubleshooting

What can you do manually to help stave off fragmentation problems? A few suggestions include:

- Set PCTINCREASE to zero for all tablespaces and objects to promote same-sized extents.
- Specify equal-sized allotments for your INITIAL and NEXT object storage parameters.
- Group objects with like growth and storage needs together in their own tablespaces.

Of all your tablespaces, you want to avoid fragmentation problems in your SYSTEM tablespace the most as this is the major hotbed tablespace for Oracle activities. The easiest way to avoid this is to not allow any user (even the default DBA ID's SYS and SYSTEM) to have access to it. There are three ways to do this:

1 Ensure no user has a DEFAULT or TEMPORARY tablespace assignment of SYSTEM.

2 Ensure no user has a quota set for SYSTEM.

3 Ensure no user has been granted the UNLIMITED TABLESPACE privilege.

Extents Deficits Tab

- Metrics

The Extents Deficits tab of the Space Detail view helps you determine if any object in your database is unable to extend into its next extent of space. Sometimes objects have NEXT extent parameters that are too large to fit into the parent tablespace's largest block of free space. For locally managed tablespaces, this is not a problem as the database handles object extent management, but for dictionary-managed tablespaces, the potential for problems exist.

By selecting a tablespace in the Extents Deficits tab, you can find out if any object extent problems are looming on the horizon of your database. The table below describes the information available on the Extents Deficits tab of the Space Detail view:

Column	Description
Owner	The user account that owns the object.
Object Name	The object name.
Object Type	The type of object.
Tablespace Name	The tablespace name.
Next Extent Size	The NEXT extent size of the object in bytes.
Max Contiguous Space	The maximum amount of contiguous space.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

You should ALTER any objects with excessive NEXT extent sizes to reduce the NEXT extent to a manageable size.

If you cannot use locally managed tablespaces to help stave off this problem, manually impose NEXT extent limits for all objects that are well within reason. Additionally, you should set the PCTINCREASE parameter for objects to zero to minimize the potential of stratospheric object growth.

Space Usage Tab

- Metrics

It is vital that a DBA know how the various accounts assigned to a database consume space. This is true in either production or development databases, because even in development databases, space can become an issue that causes major headaches. If development accounts are creating and cloning many objects without removing them, the result can be a lot of wasted space.

The Space Usage tab of the Space Detail view displays solid information to help with understanding database object ownership. The table below describes the information available on the Space Usage tab of the Space Detail view:

Column	Description
Schema	The name of the user account.
Tablespaces	The number of tablespaces in which the user currently has objects.
Spacial Objects Count	The number of space-related objects that the account owns.
Total Space Used	The amount of space that the account owns, in MB.
Percentage of Database	The database space that this account consumes, as a percentage.

Metrics

Seeing owner account with larger than expected data volumes could warrant further investigation. In addition, seeing accounts that have objects in more tablespaces than they have quota's for should also cause you to examine their account more closely to ensure the correct space privileges are in place.

Objects Page Statistics

The Objects performance category view displays the following vital Oracle objects statistics:

- Active Rollback Transactions
- Chained Row Tables
- Chained Row Fetches
- Index (ROWID) Access
- Invalid/Unusable Objects
- Locked Objects
- Long Table Scan Rows
- Object Buffer Pool Placement
- Objects With Space Deficits
- Objects With Extent Problems
- Objects Pinned

- Rollbacks at Optimal
- Rollbacks Beyond Optimal
- Total Row Fetches

Related Topics

Objects Detail

Home View Statistics

I/O Page Statistics

Memory Page Statistics

OS Page Statistics

Space Page Statistics

Top SQL Page Statistics

Users Page Statistics

Contention Statistics

Network Statistics

Object/Buffer Pool Placement

- Metrics
- Troubleshooting

Because data objects can be referenced with different types of usage patterns, Oracle8 offers the option to intelligently place objects into one of three buffer caches. The table below describes the buffer caches:

Buffer Cache	Description
KEEP	Designed to minimize misses in the buffer cache. Small objects that are frequently referenced are candidates for the KEEP buffer pool.
RECYCLE	Designed to avoid having large numbers of infrequently accessed blocks of data crowd out objects that need to be referenced in RAM, the RECYCLE pool is good for large objects that are scanned from beginning to end without the need for keeping all their data in RAM.
DEFAULT	The traditional cache for all other data objects.

NOTE: Unless you specify the KEEP or RECYCLE buffer cache, Oracle automatically places objects into the DEFAULT buffer cache.

TIP: Click this category heading to drill down to the Buffer Pool tab of the Objects Detail view.

Metrics

When looking at overall database I/O activity, you should keep an eye out for objects that you can place into a particular buffer cache. Consider using a KEEP cache for relatively small, frequently accessed tables that require fast response times. Large tables with random I/O activity and are scanned from beginning to end are good candidates for a RECYCLE cache.

Troubleshooting

Objects can be placed into different buffer pools at object creation time (using the STORAGE... BUFFER_POOL option in Oracle8 or later) or existing objects can be set into a different pool with the ALTER command.

NOTE: Tables, partitions, and indexes can be placed into the different caches.

If you just want to use the DEFAULT buffer pool and not enable any special caches, you can still encourage Oracle to keep certain objects in the cache as long as possible using the CACHE parameter. For example, issuing the command ALTER TABLE…CACHE specifies that the blocks retrieved for a table be placed at the most recently used end of the LRU list in the DEFAULT buffer cache when a full table scan is performed. You can also use the CACHE hint in SQL statements to cache a table, but used in this form, the blocks are only cached until the next time the database is shut down.

Chained Table Placement

The chained row fetches statistic represents the rows fetched that were either the chained row type or the migrated row type. The total row fetches statistic represents the total number of rows accessed by user processes.

TIP: Double-click these statistics to drill down to the Chained Tables tab of the Objects Detail view.

Metrics

If you are seeing a high percentage (30% or greater) of data access operations being satisfied by chained rows, you should take positive steps toward the elimination of chained and migrated rows from your tables.

Troubleshooting

The best way to deal with chained and migrated rows is to prevent them from occurring in the first place. The table below describes some ways to do this:

Method	Solution
Use a Large Block Size	Because chaining and migrations occur when the Oracle block is too small to hold the rows in question, make sure you are using a large enough block size for each database you create. An 8 KB or greater block size is normally recommended to help stave off chained and migrated rows.
Use Proper Values of PCTFREE for Tables	The necessary amount of percent free in a table is what helps prevent row migrations from occurring. If you have a database that houses rows with the potential to grow substantially over their initially inserted size, you can provide a liberal amount of PCTFREE for each of your tables.

If chaining or migrations are already on the rise in your database, you can take steps to counter the attack. Chaining is usually hard to resolve because rows are too big for the originally specified block size. A rebuilt database is normally the only cure for them. Migrations are another story. You can usually back up the rows in your migrated-row tables, re-create them with larger amounts of PCTFREE, and then bring the rows back in. The ANALYZE…LIST CHAINED ROWS command is useful for locating chained and migrated rows in a table.

Active Rollback Transactions

- Troubleshooting

To undo changes made to the Oracle database from within a transaction, Oracle writes data to individual rollback segments. You can also use these to maintain read consistency for multiple users of modified data. Because Rollback Segments read and write data, they can become very hot areas for I/O. This statistic is a count of how many rollback segments are actively involved in transactions for the database.

NOTE: This statistic displays on both the I/O performance category view and the Objects performance category view.

TIP: Click this statistic to drill down to the Rollback Activity tab of the I/O Detail view.

Metrics

When you drill down into this statistic and see that most or all rollback segments are involved in transactions, you should consider adding more rollback segments to the system.

Troubleshooting

If you are using Oracle8i or earlier, begin by creating new rollback segments and altering them to be online for use. Then monitor the overall active rollback ratio to see if it begins to drop.

For Oracle9i or later, consider the use of UNDO management, which is where Oracle itself automatically manages rollback segments in special UNDO tablespaces.

Index (ROWID) Accesses and Long Table Scan Rows

- Metrics
- Troubleshooting

Encouraging index access vs. full table scans on large tables is something that is always sought after. Large tables typically make up more than five data blocks, and are usually accessed better with indexes. Small tables (those with five or fewer data blocks) are surprisingly accessed more efficiently in a full table scan, because Oracle usually caches the entire table in the buffer area of the SGA. The Index (ROWID) statistic counts the number of rows in tables accessed during index access. The Table Scan Rows statistic counts the number of rows accessed for table scan operations.

Metrics

The subject of proper indexing schemes is beyond the scope of this document, but as a DBA you should keep an eye on the counts of index accesses vs. table scans. Mounting counts of large table scans are indicative of a too restrictive indexing policy, and hints that you should revisit the physical design of the database.

Troubleshooting

Here are some ways to avoid unnecessary large table scans:

- Try not to use SQL statements that include the NOT IN, NOT LIKE, < >, IS NULL operators because they typically suppress the use of indexes.
- When referencing concatenated indexes with queries, be sure the leading column in the index is used. If it is not, the index will not be used at all.
- Avoid using functions in WHERE predicates. If functions must be used, and you are using Oracle8i, investigate the use of function-based indexes.

If table scans must be used, try and make use of the following Oracle features to increase scan performance:

- Parallel query

- Partitioning
- Table CACHE parameter
- KEEP buffer pool (Oracle8)

Just adding indexes to a table does not necessarily guarantee increased performance. For normal b-tree indexes to be effective, the selectivity of the column(s) being indexed must be high. Low selectivity columns are good candidates for bitmap indexing.

With either index, make sure that the additional indexes do not overly penalize data modification performance. This can especially be the case with bitmap indexes.

Chained Row Tables

- Metrics
- Troubleshooting

In normal circumstances, a row of data should fit completely inside one Oracle block. Sometimes, however, this is not the case and the table suddenly contains chained or migrated rows (rows that span more than one data block).

Chaining occurs when a row is initially too large to fit inside one block. Two or more blocks are used by Oracle to hold the row. Migration deals with rows that have grown so much that they can no longer be contained within their original block. When this occurs, Oracle relocates the row out of its original block into another block, but leaves a pointer behind to indicate the relocation.

Both chaining and migration force Oracle to perform more than one I/O to retrieve data that could normally be obtained with a single I/O operation, with the end result being degraded performance.

> **TIP:** Click this statistic to drill down to the Chained Tables tab of the Objects Detail view.

Metrics

If the amount of chained rows in your tables exceeds 25-30%, you should take steps to eliminate the problem. Further, if the amount of chained rows accessed in your system vs. total rows accessed approaches 20-30%, you can start the process of eliminating the chained and migrated rows.

Troubleshooting

You can locate tables that contain chained rows. Once found, there are a couple of ways to reorganize tables to remove the chaining/migration problem. However, the best way to deal with chained and migrated rows is to prevent them from occurring. The table below describes two methods:

Method	Description
Use a large block size	Because chaining and migrations occur when the Oracle block is too small to hold the rows in question, make sure you are using a large enough block size for each database you create. An 8KB block size or higher is normally recommended to help stave off chained and migrated rows. If you are using Oracle9i or later, you can create special tablespaces that have larger block sizes (16-32KB) than the overall database block size and place any table that is a candidate for chained/migrated rows into them.
Use proper values of PCTFREE for tables	The necessary amount of percent free in a table helps prevent row migrations from occurring. If you have a database that houses rows with the potential to grow substantially over their initially inserted size, provide a liberal amount of PCTFREE for each of your tables.

If chaining or migrations are already on the rise in your database, you can take steps to counter the attack. Chaining is usually hard to resolve because rows are too big for the originally specified block size. A rebuilt database is normally the only cure for them. Migrations are another story. You can usually back up the rows in your migrated-row tables, re-create them with larger amounts of PCTFREE, and then bring the rows back in. The ANALYZE…LIST CHAINED ROWS command is useful for locating chained and migrated rows in a table.

Objects With Space Deficits

- Metrics
- Troubleshooting

If an existing Oracle object needs more space (due to incoming data additions, etc.) Oracle allocates new space based on the object's NEXT extent parameter. If an object's NEXT extent allocation is too large to fit into the parent tablespace's largest contiguous block of free space, and if the underlying tablespace's datafiles do not have their AUTOEXTEND property set or if they have hit their AUTOEXTEND limit, the object cannot grow in size and Oracle issues an error. For locally-managed tablespaces with generous settings for AUTOEXTEND, this problem is nonexistent as object extent management is handled by the database, but this is not the case for dictionary-managed tablespaces.

This statistic displays a count of all objects (including tables, indexes, rollback segments, etc.) that have a defined NEXT extent allocation that is larger than their tablespace's largest contiguous block of free space.

> **TIP:** Click this statistic to drill down to the Extent Deficits tab of the Space Detail view.

Metrics

You should investigate any count larger than zero, as this indicates a potential object/space problem for the database.

Troubleshooting

There are many solutions available for objects that have problematic NEXT extent sizes:

- You can ALTER the objects to reduce their NEXT extent size to an amount that is smaller than their tablespace's largest contiguous block of free space.
- If the parent tablespace suffers from heavy bubble-style fragmentation, the tablespace can be reorganized so that all pockets of free space are reclaimed.
- The parent tablespace's datafile(s) can have their AUTOEXTEND feature enabled. A side option is to have the datafile growth AUTOEXTEND option set to UNLIMITED.
- The object can be moved to a locally-managed tablespace or the parent tablespace can be converted to locally-managed (Oracle8i or later). If locally-managed tablespaces cannot be used, you can manually impose NEXT extent limits for all objects that are well within reason. In addition, you should set the PCTINCREASE parameter for objects to zero to minimize the potential of stratospheric object growth. To better control all object size/parameter allocations, you should set growth parameters for all objects at the tablespace level and not the object level.

Objects With Extent Problems

- Metrics
- Troubleshooting

Objects With Extent Problems helps you determine how many objects in your database are at or approaching maximum extents. For locally managed tablespaces, this is not a problem as the database handles object extent management, but for dictionary-managed tablespaces, the potential for problems exist.

Metrics

Use the Object Extents tab of the Object Detail view to investigate which objects are at or near max extents.

Troubleshooting

If you cannot use locally managed tablespaces to help stave off this problem, increase the number of max extents for the object.

Objects Pinned

The Library Cache is an area of memory in Oracle's shared pool that permits users to share and reuse code objects like SQL statements, PL/SQL procedures, functions, packages, and the like. To avoid expensive parse operations, the Library Cache was introduced to let Oracle cache the code objects and shared SQL so other user processes can share the same SQL or PL/SQL.

> **TIP:** Click this statistic to drill down to the Shared Pool tab of the Objects Detail view.

Metrics

Objects that have many loads into the shared pool but are not marked KEPT are ideal candidates for being permanently pinned in memory using Oracle's DBMS_SHARED_POOL utility. Doing so helps reduce expensive load operations.

Troubleshooting

Use the DBMS_SHARED_POOL utility to pin an Oracle code object in the shared pool. Keep in mind that pinning objects in memory with the DBMS_SHARED_POOL utility is only good for the duration of the current Oracle instance's life. Once you bring Oracle down, and then start back up, you need to re-pin the objects again. You can add this operation to your database maintenance plan, which is invoked on any database startup.

Invalid/Unusable Objects

The statistic combines the number of objects with a status of INVALID with the number of indexes that have a status of UNUSABLE. Objects like procedures, packages, functions, triggers, and views can become invalidated for a variety of reasons. The main cause is generally a dependent object that has been altered or removed from the system. However, other objects, like indexes, can become invalid also due to scenarios like SQL*Loader problems. If an object that has become invalid is still referenced (through an application or SQL query tool), a variety of problems can result. Sometimes Oracle reports a clear error stating the problem, while other times seemingly odd behavior is exhibited by the database.

> **TIP:** Click this statistic to drill down to the Invalid Objects tab of the Objects Detail view.

Metrics

There is no reason to have invalid objects in a production database. If your production databases have invalid objects that are no longer needed, promptly remove them from each system. Any needed objects that are indicating an invalid status should quickly be fixed before access problems develop.

It is very normal for development databases to have invalid objects because developers create, modify, and compile objects all the time. The only invalid object that really should not be present in either a development or production database is an invalid index.

Troubleshooting

If code objects have become invalidated, you can issue an ALTER … COMPILE command to see if they compile properly and become valid once again. If they do not, then check the USER_ERRORS view for any resulting error messages. Indexes can be validated once more by using the ALTER INDEX … REBUILD command.

Locked Objects

- Metrics

Locked Objects is a count of all objects on the system that currently have some form of lock against them.

> **TIP:** Click this statistic to drill down to the All Locks tab of the Locks view.

Metrics

None.

> **NOTE:** Drilling down into the count of locked objects displays detail on each object that is locked, along with the user process holding the lock and the type of lock held.

Rollback Summary

- Metrics
- Troubleshooting

These statistics display how many rollback segments are currently setting at their OPTIMAL size setting and how many have extended into a larger size. Dynamic rollback extension can take a toll on performance when rollback segments are consistently enlarged to accommodate heavy transaction loads.

> **TIP:** Double-click these statistics to drill down to the Rollback Activity tab of the I/O Detail view.

Metrics

Seeing rollback segments undergoing numerous extends and shrinks (as Oracle returns a segment to its OPTIMAL setting), as well as rollback segments having current or high watermark sizes greater than their OPTIMAL setting usually is a good indicator that they should be permanently enlarged.

Troubleshooting

For rollback segments, you can specify an OPTIMAL setting for each rollback segment. Setting OPTIMAL for the rollback gives the segment the opportunity to be shrunk back to the extent boundary where you would like to keep the rollback segment. This is periodically done by the database, but can also be accomplished by manually issuing the ALTER ROLLBACK…SHRINK command.

Objects Detail

The following tabbed pages are available on the Objects Detail page:

- Invalid Objects

- Buffer Pool
- Chained Tables
- High Watermarks
- Objects Accessed
- Object Extents
- Objects in Memory

Invalid Objects Tab

- Metrics

Objects like procedures, packages, functions, triggers, and views can become invalidated for a variety of reasons, with the main cause being a dependent object that has been altered or removed from the system. However, other objects, like indexes, can become invalid also due to scenarios like SQL*Loader problems. If an object that has become invalid is still referenced (through an application or SQL query tool), a variety of problems can result. Sometimes Oracle will report a clear error stating the problem, while other times seemingly quirky behavior will be exhibited by the database. In any event, as a DBA you should be on the lookout for objects in your database that have suddenly become invalid.

The Invalid Objects tab of the Objects Detail view displays invalid objects found in the database. The first grid displays a summary of the invalid/unusable objects. The second grid displays the details of the invalid/unusable objects. The table below describes the information available on the Invalid/Unusable Objects Summary grid on the Invalid Objects tab of the Objects Detail view:

Column	Description
Owner	The user account that currently owns invalid database objects.
Count	The number of invalid objects found in the user account.

The table below describes the information available on the Invalid/Unusable Objects Detail grid on the Invalid Objects tab of the Objects Detail view:

Column	Description
Owner	The user account that owns the objects.
Object Name	The name of the invalid object.
Object Type	The type of object: PROCEDURE, VIEW, etc.
Status	The status of the object, INVALID or UNUSABLE.
Created	The date and time stamp when the object was created.
Last DDL Date	The last structural modification date for the object.

TIP: To configure a grid to display row numbers, use the Options Editor.

Metrics

There is no reason to have invalid objects in a production database. If your production databases have invalid/unusable objects that are no longer needed, you should promptly remove them from each system. Any needed objects that are indicating an invalid status should quickly be fixed before access problems develop.

Correcting invalid objects like procedures and views often involves performing an ALTER…COMPILE operation. If the object status does not return to VALID, then further examination is warranted.

It is very normal for development databases to have invalid objects because developers will no doubt be creating, modifying, and compiling objects all the time. The only invalid object that really should not be present in either a development or production database is an invalid index.

Buffer Pool Tab

- Metrics

Because data objects can be referenced with different types of usage patterns, Oracle8 offers the option to intelligently place objects into one of three distinct buffer caches. The table below describes these buffer caches:

Buffer Cache	Description
KEEP	Minimizes misses in the buffer cache. Small, frequently referenced objects are candidates for the KEEP buffer pool.
RECYCLE	Lets you avoid having large numbers of infrequently accessed blocks of data crowd out objects that need to be referenced in RAM. The RECYCLE pool is good for large objects that are scanned from beginning to end without the need for keeping all their data in RAM.
DEFAULT	The traditional cache for all other data objects.

NOTE: Unless you specify the KEEP or RECYCLE buffer cache, Oracle automatically places objects into the DEFAULT buffer cache.

The first grid displays a summary of the objects in the buffer pool. The second grid displays the details of the objects in the buffer pool. The table below describes the information available on the Buffer Pool Objects Summary grid on the Buffer Pool tab of the Objects Detail view:

Column	Description
Object Type	The type of object (TABLE, INDEX, etc.)
Buffer Pool	The name of the buffer pool (KEEP, RECYCLE, DEFAULT).
Object Count	The number of objects in the indicated pool for the object type.

The table below describes the information available on the Buffer Pool Objects Detail grid on the Buffer Pool tab of the Objects Detail view:

Column	Description
Buffer Pool	The name of the buffer pool (KEEP, RECYCLE, DEFAULT).
Owner	The account that owns the named object.
Object Name	The name of the object.
Object Type	The type of object (TABLE, INDEX, etc.)

TIP: To configure a grid to display row numbers, use the Options Editor.

Metrics

When looking at overall database I/O activity, you should keep an eye out for objects that you can place into a particular buffer cache. Consider using a KEEP cache for relatively small, frequently accessed tables that require fast response times. Large tables with random I/O activity and are scanned from beginning to end are good candidates for a RECYCLE cache.

Objects can be placed into different buffer pools at object creation time (using the STORAGE...BUFFER_POOL option) or objects can be moved into a different pool at any time with the ALTER command.

If you just want to use the DEFAULT buffer pool and not enable any special caches, you can still encourage Oracle to keep certain objects in the cache as long as possible using the CACHE parameter. For example, issuing the command ALTER TABLE…CACHE specifies that the blocks retrieved for a table be placed at the most recently used end of the LRU list in the DEFAULT buffer cache when a full table scan is performed. You can also use CACHE hint in SQL statements to cache a table, but used in this form, the blocks will only be cached until the next time the database is shut down.

Chained Tables Tab

- Metrics

In normal circumstances, a row of data should fit completely inside one Oracle block. Sometimes, however, this is not the case and the table suddenly finds itself containing chained or migrated rows, which are rows that span more than one data block.

Chaining occurs when a row is initially too large to fit inside one block. Two or more blocks are used by Oracle to hold the row. Migration deals with rows that have grown so much that they can no longer be contained within their original block. When this occurs, Oracle relocates the row out of its original block into another block, but leaves a pointer behind to indicate the relocation. Both chaining and migration force Oracle to perform more than one I/O to retrieve data that could normally be obtained with a single I/O operation, resulting in degraded performance.

The Chained Tables tab of the Objects Detail view displays tables that currently have chained rows. Note that for truly accurate data to be returned, the ANALYZE command needs to be routinely inside the database. This ensures that the data dictionary is up to date with respect to object demographics.

The table below describes the information available on the Chained Tables tab of the Objects Detail view:

Column	Description
Schema	The user account that owns the chained tables.
Table Name	The table name.
Rows	The number of rows that reside inside the table.
Chained Rows	The number of chained rows that reside inside the table.
Percent Chained	The percentage of chained rows to total rows in the table.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

How bad should chaining or migration get before you take action with a table? If the amount of chained rows in your tables exceeds 25-30%, you should take steps to begin eliminating the problem. Further, if the amount of chained rows accessed in your system vs. total rows accessed approaches 20-30%, you can start the process of eliminating the chained and migrated rows.

The best way to deal with chained and migrated rows is to prevent them from occurring in the first place. The table below describes some ways to do this:

Method	Solution
Use a Large Block Size	Because chaining and migrations occur when the Oracle block is too small to hold the rows in question, make sure you are using a large enough block size for each database you create. An 8 KB or greater block size is normally recommended to help stave off chained and migrated rows.

Method	Solution
Use Proper Values of PCTFREE for Tables	The necessary amount of percent free in a table is what helps prevent row migrations from occurring. If you have a database that houses rows with the potential to grow substantially over their initially inserted size, you can provide a liberal amount of PCTFREE for each of your tables.

If chaining or migrations are already on the rise in your database, you can take steps to counter the attack. Chaining is usually hard to resolve because rows are too big for the originally specified block size. A rebuilt database is normally the only cure for them. Migrations are another story. You can usually back up the rows in your migrated-row tables, re-create them with larger amounts of PCTFREE, and then bring the rows back in.

High Watermarks Tab

- Metrics

Tables that are the victim of much insert-delete activity can become performance problems due to their block high watermarks. A table's high watermark equals the last block in the table that was used to hold data. The problem is that the high watermark is not reset by DELETE activity, so it is possible for a table to have absolutely no data in it, but contain a high watermark that is many blocks high. When such a table is scanned, Oracle reads up to the high watermark even if no rows exist in the table at all. This can make for some unnecessarily large scan times.

The table below describes the information available on the High Watermarks tab of the Objects Detail view:

Column	Description
Owner	The user account that owns the tables.
Table Name	The name of the table.
Size (bytes)	The physical size of the table, in bytes.
Rows	The number of rows occupying the table.
Blocks	The number of blocks the table consumes.
Empty Blocks	The number of blocks assigned to the table that contain no data.
High Watermark	The last block in the table that was used to hold data.

NOTE: For truly accurate data to be returned, the ANALYZE command needs to be routinely run on the viewed tables. This ensures that the data dictionary is up to date with respect to object demographics.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Red flags should go up to you if you observe a table that contains no rows (or few rows) but has a high watermark that is nonzero for no rows, and much above zero for a few. Again, to effectively detect a table's high watermark, you must make timely use of the ANALYZE command to update the data dictionary. Doing so ensures a fresh supply of statistics available to analyze used and free data blocks in a table.

Objects Accessed Tab

- Metrics

There are two things to consider regarding the performance of objects and Oracle memory use. First, because data can be accessed many times faster in RAM than from disk, you should keep often-referenced data from your objects in memory as much as possible. It is also wise to keep object definitions in memory so that references to them are quick.

Second, to avoid expensive parsing operations and continuous loads into the Oracle shared pool, you should do everything possible to encourage code reuse and ensure that parsed code is ready and available in the library cache.

The Objects Accessed tab of the Objects Detail view displays what objects are currently being accessed in memory and users who are using them. The table below describes the information available on the Objects Accessed tab of the Objects Detail view:

Column	Description
Owner	The user account that owns the object.
Object name	The name of the object in memory.
Type	The type of object (TABLE, CURSOR, VIEW, etc.)
Accessing User	The user who is currently using the object.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Once you have an idea of what objects your users are most often after, you can do a couple of things to help their access process along.

With respect to data objects, you can use the Oracle8 concept of the KEEP buffer cache to force Oracle to hold often referenced data. The KEEP buffer cache is ideal for holding small lookup tables and the like. If you have an earlier version of Oracle or don't want to divide your current buffer cache, you can alternatively use the CACHE table attribute to encourage Oracle to keep data blocks of cached tables at the most recently used end of the LRU buffer cache chain.

Objects that have many loads but are not marked KEPT in the shared pool are ideal candidates for being permanently pinned in memory using the DBMS_SHARED_POOL utility. Doing so helps reduce expensive load operations. Keep in mind that pinning objects in memory with the DBMS_SHARED_POOL utility is only good for the duration of the current Oracle instance's life. Once Oracle is brought down and started back up, re-pin the objects.

Object Extents Tab

- Metrics

Object fragmentation results when objects consume multiple extents of space in a tablespace rather than a single block of space. Although performance problems with respect to object fragmentation are not as severe as they were in older versions of Oracle, response-time penalties can still be chalked up to this situation. When multiple extents exist for an object, the amount of time it takes Oracle to scan it can be longer than if the object was made up of only one extent. This typically holds true when extents are scattered on different parts of the physical server disk. In addition, a performance hit is taken each time an object must extend into another extent of space.

The Object Extents tab of the Objects Detail view displays objects whose extent count has exceeded a user-suggested numerical limit. If objects are found, Performance Center displays them on the this tab. The table below describes the information available on the Object Extents tab of the Objects Detail view:

Column	Description
Owner	The user account that owns the object.
Object Name	The name of the object.
Object Type	The type of object (TABLE, INDEX, etc.)
Extents	The number of extents for the object.
Limit	The MAXEXTENTS limit imposed on the object by the DBA.
Pct/Limit	How close the object is to reaching its extent limit, in percentage.

NOTE: For more information see, Object Extents Tab.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

When is object fragmentation a problem for an object? Opinions on this vary widely. As a general rule of thumb, if an object is exhibiting response time degradations, and no other factor can be attributed to the slowdown, examine the object to see how badly fragmented it is. If extent numbers exceed 20, you should consider reorganizing the object back into a single extent of space. This rule of thumb does not apply to objects assigned to locally managed tablespaces as extent fragmentation is expected and not a performance issue.

The best preventive for this problem is specifying the correct allocation of space for the object in the first place, but what can you do if you have objects with high numbers of extents? For tables and indexes you have three options:

1. Use Oracle's export/import utility to export, drop, and import the fragmented objects back into the database with the export parameter COMPRESS=Y. This brings the objects back into the database with one large extent. Make sure that large enough chunks of available free space exist to accept the object back in, or you may experience space allocation errors.
2. With Oracle8, you can use the ALTER TABLE…MOVE command to reorganize a table back into one extent in a very safe and efficient manner.
3. Use ALTER INDEX…REBUILD to reorganize indexes that have moved into multiple extents.

For rollback segments, you can specify an OPTIMAL setting for each rollback segment. Setting OPTIMAL for the rollback gives the segment the opportunity to be shrunk back to the extent boundary where you would like to keep the rollback segment. This is periodically done by the database, but can also be accomplished by manually issuing the ALTER ROLLBACK…SHRINK command.

Objects in Memory Tab

- Metrics

The Objects in Memory tab of the Objects Detail view shows objects currently in the library cache. This view lets you see exactly which objects are in the shared pool as well as their resource usage and activity levels.

The table below describes the information available on the Objects in Memory tab of the Objects Detail view:

Column	Description
Owner	The user account that owns the object.
Object Name	The name of the object.
Type	The type of object: INDEX, TABLE, CLUSTER, VIEW, SET, SYNONYM, SEQUENCE, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, TRIGGER, CLASS, OBJECT, USER, or DBLINK.
Sharable Memory	The amount of memory consumed by the object in the shared pool.
Loads	The number of times the object has been loaded into the cache. This count increases when an object has been invalidated.
Executions	The number of times that the object has been executed by a session thread.
Locks	The number of users actively locking the object.
Pins	The number of users actively pinning the object.
Kept	Whether or not the object has been pinned in memory with the DBMS_SHARED_POOL package.

NOTE: This information is available on both the Objects in Memory tab of the Memory Detail view and the Objects in Memory tab of the Objects Detail view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

If you see objects with large numbers of loads, this could signal that the objects are being frequently by user sessions and can age out of the cache prematurely. To fix this you could increase the size of the shared pool and/or pin the objects with the DBMS_SHARED_POOL package so they do not age out of the library cache.

OS Page Statistics

In many scenarios, an optimally tuned database may not perform well because there are constraints imposed by the system where the database is running. These constraints may include processes competing with the database sever for resources (CPU, I/O, or Memory), a slow CPU, insufficient or slow I/O devices, and insufficient memory. The OS Statistics page of Performance Center lets you examine operating system metrics for the following platforms:

- AIX
- HP-UX

  NOTE: To view processor info and swap disk info on an HP-UX box, you need to login as ROOT in the OS login.

- Linux
- Solaris
- Unix
- Windows XP, 2000, and NT

CAUTION: The OS statistics are not available from the Web Client.

- Average Disk Queue
- Network Output Queue (Windows)
- Network Queue (Solaris)
- Page Faults/Sec
- Processor Queue
- Processor Speed
- Processor
- Available Paged Memory (Windows)
- Available Physical Memory
- Available Swap Memory (AIX, HP-UX, Linux, Solaris, Unix)
- Total Paged Memory (Windows)
- Total Physical Memory
- Total Swap Memory (AIX, HP-UX, Linux, Solaris, Unix)
- Free Disk Space
- Total Disk Space
- Used Disk Space
- Number of Logins
- Number of Processes
- Number of Processors
- Top CPU Process
- Top I/O Process
- Top Memory Process

Processor Time

The Processor Time statistic indicates the percentage of time the processor is working. This counter is a primary indicator of processor activity.

Metrics

If your computer seems to be running sluggishly, this statistic could be displaying a high percentage.

Troubleshooting

Upgrade to a processor with a larger L2 cache, a faster processor, or install an additional processor.

Processor Speed

The Processor Speed statistic displays the speed of the active processor in MHz. The speed is approximate.

Processor

The Processor Statistic displays the type of processor currently in use, for example, GenuineIntel.

Disk Time

The Disk Time statistic is the percentage of elapsed time that the selected disk drive/device was busy servicing read or write requests.

Metrics

You should avoid consistently seeing values for this statistic greater then 90%.

Troubleshooting

Add more disk drives and partition the files among all of the drives.

Load Average

The Load Average statistic represents the system load averages over the last 1, 5, and 15 minutes.

Metrics

High load averages usually mean that the system is being used heavily and the response time is correspondingly slow.

Paged Memory Used

The Paged Memory Used statistic is the ratio of Commit Memory Bytes to the Commit Limit. Committed memory is where memory space has been reserved in the paging file if it needs to be written to disk. The commit limit is determined by the size of the paging file. As the paging file increases, so does the commit limit.

NOTE: This statistic is available for the Windows platform.

Metrics

This value displays the current percentage value only and not an average. If the percentage of paged memory used is above 90%, you may be running out of memory.

Troubleshooting

Increase the size of page file.

Number of Processors

This statistic displays the number of processors currently in use.

Swap Memory Used

The Swap Memory Used statistic is the percentage of swap space currently in use.

Metrics

If the percentage of swap memory used is above 90%, you may be running out of memory.

Troubleshooting

Increase the size of your swap files.

Average Disk Queue

The Average Disk Queue statistic is the average number of both read and write requests that were queued for the selected disk during the sample interval.

Metrics

This metric is useful in identifying I/O related bottlenecks. If the disk queue lengths for certain disks are consistently much higher than others, you may need to redistribute the load among available disks. If the disk queues lengths for all disks are consistently large, and you see a high amount of I/O activity, your disks may be inefficient.

Troubleshooting

Some things you can do if you have problems with this statistic include:

- Redistribute the data on the disk with the large average disk queue to other disks.
- Upgrade to faster disk(s).

Page Faults/Sec

The Page Faults/Sec statistic is the overall rate faulted pages are handled by the processor. It is measured in numbers of pages faulted per second. A page fault occurs when a process requires code or data that is not in its working set. This counter includes both hard faults and soft faults.

Metrics

This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

If the number of page faults remains consistently high, you can check with your Windows System Administrator for further investigation. Often, large numbers of page faults are not a problem so long as they are soft faults. However, hard faults, that require disk access, can cause delays.

Processor Queue

The Processor Queue Length statistic is the number of threads in the processor queue.

Metrics

Unlike the disk counters, this counter shows ready threads only, not threads that are running. There is a single queue for processor time even on computers with multiple processors. Therefore, if a computer has multiple processors, you need to divide this value by the number of processors servicing the workload. A sustained processor queue of less than 10 threads per processor is normally acceptable, dependent of the workload.

Troubleshooting

A sustained high value in the Processor Queue could indicate that a processor bottleneck has developed due to threads of a process requiring more process cycles than are available. If this is the case, you should look at installing a faster (or an additional) processor.

Network Output Queue/Network Queue

The Network Output Queue Length statistic is the number of threads in the processor queue.

NOTE: The name of this statistic depends on the platform of the operating system.

Metrics

Unlike the disk counters, this counter shows ready threads only, not threads that are running. There is a single queue for processor time even on computers with multiple processors. Therefore, if a computer has multiple processors, you need to divide this value by the number of processors servicing the workload. A sustained processor queue of less than 10 threads per processor is normally acceptable, dependent of the workload.

Troubleshooting

A sustained high value in the Processor Queue Length could indicate that a processor bottleneck has developed due to threads of a process requiring more process cycles than are available. If this is the case, you should look at installing a faster (or an additional) processor.

Available Physical Memory

The Available Physical Memory statistic represents the amount of RAM available to all processes.

Metrics

This counter displays the last observed value only and not an average. Use this value with the Total physical memory and paging metrics (Memory details page). If the available physical memory is very small compared to this value, and the paging activity is high, your system may be running low on memory.

Troubleshooting

Some things you can do if you have problems with this statistic include:

- Check the running processes to see if there are any memory leaks.
- Stop any services that are not required.
- Install additional RAM.

Available Paged Memory

The Available Paged Memory statistic shows the amount of virtual memory available for the processes.

NOTE: This statistic is available for the Windows platform.

Metrics

If the available virtual memory is less than 10% of the total virtual memory, your system may run out of memory.

Troubleshooting

Increase the size of page file.

Available Swap Memory

The Available Swap Memory statistic represents the amount of virtual memory available for the processes.

Metrics

If the available Available Swap Memory is less than 10% of the total Swap Memory, your system may run out of memory.

Troubleshooting

Increase the size of swap files.

Total Physical Memory

The Total Physical Memory statistic shows the amount of physical memory installed on your computer.

Metrics

This is an informational metric and displays the total amount installed on the machine. Use this value with the available physical memory and paging metrics (Memory details page). If the available physical memory is very small compared to this value, and the paging activity is high, your system may be running low on memory.

Total Paged Memory/Total Swap Memory

The Total Paged Memory statistic shows the maximum amount of virtual memory available to all processes.

NOTE: The name of this statistic depends on the platform of the operating system.

Metrics

It is recommended that this value is between 1.5 to 3 times the amount of RAM on the system.

Used Disk Space

The Used Disk Space statistic shows the amount of allocated space, in megabytes on all logical disk drives.

Troubleshooting

There are many things a DBA can do to ensure that a database does not encounter a space problem due to physical space limitations:

- If a database currently resides on a disk that has little free space, you can add more files to the database. Of course, you should add the new files to other physical hard disks that can accommodate a growing database.
- You should examine hard disks with shrinking disk space to see if you can relocate or delete files to allow more free space.

Total Disk Space

Total Disk Space displays the total allocated and unallocated space, in megabytes on all logical disk drives.

Troubleshooting

There are many things a DBA can do to ensure that a database does not encounter a space problem due to physical space limitations, here are two:

1. If a database currently resides on a disk that has little free space, you can add more files to the database. Of course, you should add the new files to other physical hard disks that can accommodate a growing database.
2. You should examine hard disks with shrinking disk space to see if you can relocate or delete files to allow more free space.

Free Disk Space

The Free Disk Space statistic shows the unallocated space, in megabytes on all logical disk drives.

Troubleshooting

There are many things a DBA can do to ensure that a database does not encounter a space problem due to physical space limitations:

- If a database currently resides on a disk that has little free space, you can add more files to the database. Of course, you should add the new files to other physical hard disks that can accommodate a growing database.
- You should examine hard disks with shrinking disk space to see if you can relocate or delete files to allow more free space.

Top Memory Process

Top Memory Process shows the current process that is consuming the most amount of memory. The information displayed is dependent on the platform of the operating system. Information displayed includes the name of the process, process ID, amount of memory consumed expressed in KB, amount of CPU expressed as a percentage, the amount of Major Page Faults, and the amount of I/O expressed in KB/sec.

Metrics

If you are running out of memory on the system, this is a quick way to identify the top memory user. If the displayed process is using a significant portion of the total memory, it could be causing the memory issues.

Processes Overview

The Processes Overview of the OS Summary includes the following sections:

- Top CPU Process
- Top I/O Process
- Top Memory Process

Top CPU Process

Top CPU Process shows the current process that is consuming the most amount of CPU. The information displayed is dependent on the platform of the operating system. Information displayed includes the name of the process, process ID, amount of memory consumed expressed in KB, amount of CPU expressed as a percentage, the amount of Major Page Faults, and the amount of I/O expressed in KB/sec.

Metrics

If the amount of CPU time used by this process is close to 100% and the CPU usage is very high, this process may be the bottleneck on the server.

Troubleshooting

Investigate the process further to see if it is in an inconsistent state. Also, look at minimum requirements for CPU speed for the process. You may need to upgrade your CPU.

Top I/O Process

The Top I/O Process statistic shows the current process that is consuming the most amount of CPU. The information displayed is dependent on the platform of the operating system. Information displayed includes the name of the process, process ID, amount of memory consumed expressed in KB, amount of CPU expressed as a percentage, the amount of Major Page Faults, and the amount of I/O expressed in KB/sec.

Top Memory Process

The top I/O process identifies the Oracle process that currently is using the highest percentage of memory in the database.

Metrics

To obtain more details on the top memory process, locate the SID in the Top Sessions grid and drill down to obtain more granular information.

Number of Logins

This statistic displays the total number of logins on the server.

Number of Processes

This statistic displays the total number of processes on the server.

CPU Tab

The CPU tab of the OS Detail includes the following sections:

- Context Switches/Sec
- CPU Utilization
- Interrupts/Sec
- Processor Queue Length

CPU Utilization

The CPU Utilization section includes the following information:

- % Privileged Time
- % User Time

% Privileged Time

The % Privileged Time statistic is the percentage of elapsed time that the process threads spent executing code in privileged mode.

> **NOTE:** For Windows systems, when a Windows system service is called, the service will often run in privileged mode to gain access to system-private data. Such data is protected from access by threads executing in user mode. Calls to the system can be explicit or implicit, such as page faults or interrupts. These kernel commands, are considered privileged to keep the low-level commands executing and prevent a system freeze. Unlike some early operating systems, Windows uses process boundaries for subsystem protection in addition to the traditional protection of user and privileged modes. Some work done by Windows on behalf of the application might appear in other subsystem processes in addition to the privileged time in the process.

Metrics

The ideal range should be 0-40% (less than 40% indicates excessive system activity).

Troubleshooting

If your CPU consistently runs at less than 40% you may need to upgrade your system to include a faster processor(s).

% User Time

The % User Time statistic is the percentage of elapsed time the processor spends in the user mode. User mode is a restricted processing mode designed for applications, environment subsystems, and integral subsystems. The alternative, privileged mode, is designed for operating system components and allows direct access to hardware and all memory. The operating system switches application threads to privileged mode to access operating system services. This counter displays the average busy time as a percentage of the sample time.

Metrics

If the Privileged Time is high in conjunction with Physical Disk Reads, consider upgrading the disk I/O subsystem.

Interrupts/Sec

The Interrupts/Sec statistic is the average rate, in incidents per second, at which the processor received and serviced hardware interrupts. It does not include deferred procedure calls (DPCs), which are counted separately. This value is an indirect indicator of the activity of devices that generate interrupts, such as the system clock, the mouse, disk drivers, data communication lines, network interface cards, and other peripheral devices. These devices normally interrupt the processor when they have completed a task or require attention. Normal thread execution is suspended. The system clock typically interrupts the processor every ten milliseconds, creating a background of interrupt activity. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Metrics

The ideal rage should be 0-5000. A number greater then 5000 indicates possible excessive hardware interrupts; justification is dependent on device activity.

Context Switches/Sec

The Context Switches/Sec section shows the combined rate at which all processors on the computer are switched from one thread to another. Context switches occur when a running thread voluntarily relinquishes the processor, is preempted by a higher priority ready thread, or switches between user-mode and privileged (kernel) mode to use an Executive or subsystem service.

Metrics

The ideal range should be between 0-10,000. GA number greater then 10,000 may indicate too many threads contending for resources.

Processor Queue Length

The Processor Queue Length statistic is the number of threads in the processor queue. There is a single queue for processor time even on computers with multiple processors.

> **NOTE:** For Windows systems, unlike the disk counters, this counter shows ready threads only, not threads that are running.

Metrics

A sustained high value in the Processor Queue Length could indicate that a processor bottleneck has developed due to threads of a process requiring more process cycles than are available. If this is the case, you should look at installing a faster (or an additional) processor.

Processes Tab

The Processes tab of the OS Detail page succinctly communicates the general overall performance levels of processes. The columns available in this table depend on the platform of operating system. The table below describes the information available in the table on this tab:

Column	Description
Process	The name of the process.
User	The user of the process.
ID	The ID Process is the unique identifier of this process. ID Process numbers are reused, so they only identify a process for the lifetime of that process.
CPU	The CPU is the percentage of elapsed time that all of process threads used the processor to execution instructions.
User Mode	The User Mode is the percentage of elapsed time that the process threads spent executing code in user mode.
Memory **WINDOWS ONLY**	Memory is the current size, in bytes, of the virtual address space the process is using. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. Virtual space is finite, and the process can limit its ability to load libraries.
Memory (MB)	Memory is the current size, in bytes, of the virtual address space the process is using. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. Virtual space is finite, and the process can limit its ability to load libraries.
Memory	Memory is the percentage of the memory used of the total memory.
Active Memory	Active Memory is the amount of committed virtual memory, in bytes for this process. Active memory is the physical memory which has space reserved on the disk paging file(s). There can be one or more paging files on each physical drive. This counter displays the last observed value only; it is not an average.

Column	Description
I/O Data	The rate at which the process is reading and writing bytes in I/O operations. This counter counts all I/O activity generated by the process to include file, network and device I/Os.
Elapsed Time	The total elapsed time, in seconds, that this process has been running.
Thread Count	The number of threads currently active in this process. An instruction is the basic unit of execution in a processor, and a thread is the object that executes instructions. Every running process has at least one thread.
Handle Count	The total number of handles currently open by this process. This number is equal to the sum of the handles currently open by each thread in this process.
Priority	The current base priority of this process. Threads within a process can raise and lower their own base priority relative to the process' base priority.
Creating Proc ID	The Creating Process ID value is the Process ID of the process that created the process. The creating process may have terminated, so this value may no longer identify a running process.
Page Faults/Sec	Page Faults/Sec is the rate at which page faults by the threads executing in this process are occurring. A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory. This may not cause the page to be fetched from disk if it is on the standby list and hence already in main memory, or if it is in use by another process with whom the page is shared.
Page File	Page File is the current number of kilobytes that this process has used in the paging file(s). Paging files are used to store pages of memory used by the process that are not contained in other files. Paging files are shared by all processes, and the lack of space in paging files can prevent other processes from allocating memory.
Private	Private is the current size, in kilobytes, of memory that this process has allocated that cannot be shared with other processes.

I/O Tab

The table below describes the information available in this section:

Column	Description
Disk	The disk number assignment.
Reading (KB/s)	The amount of bytes read from the device.
Writing (KB/s)	The amount of bytes written to the device.
Disk Read Time	Disk Read Time is the percentage of elapsed time that the selected disk drive was busy servicing read requests.
Disk Write Time	Disk Write Time is the percentage of elapsed time that the selected disk drive was busy servicing write requests.
Disk Time	Disk Time is the percentage of elapsed time that the selected disk was busy servicing requests.
Avg. Read Queue	Avg. Disk Read Queue Length is the average number of read requests that were queued for the selected disk during the sample interval.
Avg. Write Queue	Avg. Disk Write Queue Length is the average number of write requests that were queued for the selected disk during the sample interval.
Disk Reads/Sec	Disk Reads/Sec is the rate of read operations on the disk.
Disk Writes/Sec	Disk Writes/Sec is the rate of write operations on the disk.

Memory Tab

The Memory tab of the OS Detail page includes the following sections:

- Cache Efficiency
- Free Physical
- Free Paged
- Paging Activity
- Page Faults
- Total Physical
- Total Paged

Paging Activity

The Paging Activity section includes the following statistics:

- Pages Input/Sec
- Pages Output/Sec

Pages Input/Sec

The Pages Input/Sec statistic is the number of pages read from disk to resolve hard page faults. Hard page faults occur when a process requires code or data that is not in its working set or elsewhere in physical memory, and must be retrieved from disk.

Metrics

This value was designed as a primary indicator of the kinds of faults that cause system-wide delays. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) and in non-cached mapped memory files. This counter counts numbers of pages, and can be compared to other counts of pages, such as Memory: Page Faults/sec, without conversion. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

Although it never hurts to have as much physical memory as your system can handle, there are some things you can check within your system to alleviate the memory bottleneck.

- Check to see if you have any drivers or protocols that are running but not being used. They use space in all memory pools even if they are idle.
- Check to see if you have additional space on your disk drive that you could use to expand the size of your page file. Normally, the bigger the initial size of your page file, the better, in performance terms.

Pages Output/Sec

The Pages Output/Sec statistic is the number of pages written to disk to free up space in physical memory. Pages are written back to disk only if they are changed in physical memory. A high rate of pages output might indicate a memory shortage.

Metrics

Windows NT writes more pages back to disk to free up space when low in physical memory. This counter counts numbers of pages, and can be compared to other counts of pages, without conversion. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

Although it never hurts to have as much physical memory as your system can handle, there are some things you can check within your system to alleviate the memory bottleneck.

- Check to see if you have any drivers or protocols that are running but not being used. They use space in all memory pools even if they are idle.
- Check to see if you have additional space on your disk drive that you could use to expand the size of your page file. Normally, the bigger the initial size of your page file, the better, in performance terms.

Free Physical

The Free Physical statistic is the amount of physical memory that is uncommitted.

Metrics

None.

Free Paged

The Free Paged statistic is the amount of uncommitted virtual memory.

Metrics

None.

Total Physical

The Total Physical statistic is the total physical memory available.

Metrics

None.

Total Paged

The Total Paged statistic is the amount of total virtual memory, in bytes. Used Memory is the physical memory that has space reserved on the disk paging file(s). There can be one or more paging files on each physical drive.

Metrics

None.

Page Faults/Sec

The Page Faults/Sec statistic is the overall rate faulted pages are handled by the processor. It is measured in numbers of pages faulted per second. A page fault occurs when a process requires code or data that is not in its working set. This counter includes both hard faults and soft faults.

Metrics

This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

If the number of page faults remains consistently high, you can check with your Windows System Administrator for further investigation. Often, large numbers of page faults are not a problem so long as they are soft faults. However, hard faults, that require disk access, can cause delays.

Cache Efficiency

The Cache Efficiency section of the Memory tab succinctly communicates the general overall performance levels of the server's memory. The following statistics are available in this section:

- Copy Read Hits%
- Data Map Hits%
- MDL Read Hits%
- Pin Read Hits%

Copy Read Hits %

The Copy Read Hits % statistic is the percentage of cache copy read requests that hit the cache and does not require a disk read to provide access to the page in the cache.

Metrics

When the page is pinned in the memory, the page's physical address in the file system cache will not be altered. A copy read is a file read operation where a page in the cache is copied to the application's buffer. Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate

Data Map Hits %

The Data Map Hits % statistic is the percentage of data maps in the file system cache that could be resolved without having to retrieve a page from the disk.

Metrics

Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate.

MDL Read Hits %

The MDL Read Hits % statistic is the percentage of Memory Descriptor List Read requests to the file system cache that hit the cache and does not require disk access to provide memory access to the pages in the cache.

Metrics

Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate.

Pin Read Hits %

The Pin Read Hits % statistic is the percentage of pin read requests that hit the file system cache and does not require a disk read in order to provide access to the page in the file system cache.

Metrics

Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate.

Space Tab

The Space tab of the OS Detail page includes the following sections:

- Disk Space Free
- Disk Space Detail

Disk Space Free

The Disk Space Free metric displays the amount of free disk space in megabytes.

Metric

None.

Disk Space Detail

The Disk Space Detail section of the Space tab succinctly communicates the general overall performance levels of the server's disks and space allotment. The table below describes the statistics in this section:

Statistic	Description
Partition	The drive letter of the disk.
Local Filesystem	The name of the file system.
Type	The type of file system.
Total Space	Total size of the disk/device's capacity expressed in MBs.
Used Space	Amount of MBs currently allocated on the particular disk/device.
Free Space	Amount of MBs currently unallocated and free on the particular disk/device.

Statistic	Description
Capacity	The percentage of space used on the device.
Mounted On	The mount point of the device.

Network Tab

The Network tab of the OS Detail page succinctly communicates the general overall performance levels of the server's networking. The statistics available in this section depend on the platform of operating system. The table below describes the information available in this section:

Column	Description
Network Interface	The name of network interface.
INET Address/Address	The IP address assigned to the network interface.
Pkts Sent/Sec	The number of packets sent per second.
Pkts Received/Sec	The number of packets received per second.
Sent (KB/Sec)	The number of bytes sent per second.
Received (KB/Sec)	The number of bytes received per second.
Out Pkts Discarded	The number of outbound packets discarded.
In Pkts Discarded	The number of inbound packets discarded.
Out Pkt Errors	The number of outbound packet errors.
In Pkt Errors	The number of inbound packet errors.
Queue Length	The queue length.
Collisions	The number of collisions.
Packets Discarded	The number of packets discarded.

Contention Statistics - Oracle

The Contention performance category view displays the following vital Oracle contention statistics:

- Buffer Busy Wait Ratio
- Enqueue Waits
- Free List Waits
- Latch Immediate Miss Ratio
- Latch Miss Ratio
- Latch Sleep Ratio
- Max Query Slaves
- Parallel Query Busy Ratio

- Prolonged Lock Waits
- Redo Log Space Requests
- Redo Log Wait Time
- Rollback Contention Ratio
- Sessions in Buffer Busy Waits
- Sessions Waiting for Latches
- Used Query Slave
- Users Blocked
- Users Waiting (General)

Latch Miss Ratio

- Metrics
- Troubleshooting

The latch miss ratio defines the number of times a process obtained a willing-to-wait latch vs. missing the attempt. Latches protect the many memory structures in Oracle's SGA. They ensure that one and only one process at a time runs or modifies any memory structure at the same instant. Latches are much more restrictive than locks, which at least allow for some collective user interaction. They have no queuing mechanism, and therefore, you either get the latch or you are forced to continually retry.

> **TIP:** Click this statistic to drill down to the System Waits tab of the Contention Detail view.

Metrics

If the latch miss ratio exceeds 1%, you should take action to resolve the amount of latch contention occurring.

Troubleshooting

You should examine the details regarding the latch contention. Increasing the shared_pool_size can also assist in resolving latch problems. Here are a few specialized cases of latch contention that you can easily rectify:

Latch Contention Case	Description	Resolution
Cache buffer chain latch	This latch is responsible for protecting paths to database block buffers in the buffer cache. Very high I/O loads tend to cause contention for this latch.	You can alleviate contention somewhat by adding more buffers to the cache (through the db_block_buffers parameter) or by adding more LRU latch chain latches with the db_block_lru_latches parameter.
Library cache latches	Protects cached SQL statements in the library cache area of the Oracle shared pool. Contention for this latch is the usual result of literals being used for SQL statements instead of bind variables.	If possible, make SQL more reusable by using bind variables instead of literals. Doing so should alleviate contention for the library cache latch."

Other routine latch contention problems used to include the redo allocation and redo copy latches, however in Oracle 8.1.5 or later these are generally obsolete.

Latch Immediate Miss Ratio

The latch immediate miss ratio defines the number of times a process obtained a not-willing-to-wait latch vs. missing the attempt. Latches protect the many memory structures in Oracle's SGA. They ensure that one and only one process at a time runs or modifies any memory structure at the same instant. Latches are much more restrictive than locks, which at least allow some collective user interaction. They have no queuing mechanism, and therefore, you either get the latch or you are forced to continually retry.

> **TIP:** Click this statistic to drill down to the Latch Detail tab of the Contention Detail view.

Metrics

If the latch immediate miss ratio exceeds 1%, you should take action to resolve the amount of latch contention occurring.

Troubleshooting

You should examine the details regarding the latch contention. Increasing the shared_pool_size can also assist in resolving latch problems. Here are a few specialized cases of latch contention that you can easily rectify:

Latch Contention Case	Description	Resolution
Cache buffer chain latch	This latch is responsible for protecting paths to database block buffers in the buffer cache. Very high I/O loads tend to cause contention for this latch.	You can alleviate contention somewhat by adding more buffers to the cache (through the db_block_buffers parameter) or by adding more LRU latch chain latches with the db_block_lru_latches parameter.
Library cache latches	Protects cached SQL statements in the library cache area of the Oracle shared pool. Contention for this latch is the usual result of literals being used for SQL statements instead of bind variables.	If possible, make SQL more reusable by using bind variables instead of literals. Doing so should alleviate contention for the library cache latch."

Other routine latch contention problems previously included the redo allocation and redo copy latches. In Oracle 8.1.5 or later these are generally obsolete.

Latch Sleep Ratio

Latches protect the many memory structures in Oracle's SGA by ensuring that one and only one process at a time runs or modifies any memory structure at the same instant. Latches are much more restrictive than locks, which at least allow for some collective user interaction. They have no queuing mechanism, and therefore, you either get the latch or you are forced to continually retry.

A sleep indicates that a latch could not be obtained for a process, and that Oracle continues to retry. A high ratio indicates that many processes had to sleep multiple times before obtaining a requested latch.

> **TIP:** Click this statistic to drill down to the Latch Detail tab of the Contention Detail view.

Metrics

You should keep sleeps as low as possible. If the overall sleep ratio exceeds 1%, you should take action to resolve the amount of latch contention that is occurring.

Troubleshooting

You should examine the details regarding the latch contention. Increasing the shared_pool_size can also assist in resolving latch problems. Here are a few specialized cases of latch contention that you can easily rectify:

Latch Contention Case	Description	Resolution
Cache buffer chain latch	This latch is responsible for protecting paths to database block buffers in the buffer cache. Very high I/O loads tend to cause contention for this latch.	You can alleviate contention somewhat by adding more buffers to the cache (through the db_block_buffers parameter) or by adding more LRU latch chain latches with the db_block_lru_latches parameter.
Library cache latches	Protects cached SQL statements in the library cache area of the Oracle shared pool. Contention for this latch is the usual result of literals being used for SQL statements instead of bind variables.	If possible, make SQL more reusable by using bind variables instead of literals. Doing so should alleviate contention for the library cache latch."

Other routine latch contention problems previously included the redo allocation and redo copy latches. In Oracle 8.1.5 or later these are generally obsolete.

Buffer Busy Wait Ratio

- Metrics
- Troubleshooting

TIP: Click this statistic to drill down to the Buffer Busy Waits tab of the Contention Detail view.

Metrics

Buffer busy waits normally center around contention for rollback segments, too small an INITRANS setting for tables, or insufficient free lists for tables.

Troubleshooting

On Oracle8i or earlier, the remedy for each situation would be increasing the number of rollback segments, or altering tables to have larger settings for INITRANS to allow for more transactions per data block, and more free lists.

For Oracle9i or later, you can use the automatic segment management feature in Oracle9i locally-managed tablespaces to help make free list problems a thing of the past. Using an UNDO tablespace in 9i or later can help remedy any rollback contention problem.

You can also obtain which objects have actually experienced buffer busy waits in Oracle9i or later by querying the sys.v_$segment_statistics. This view is not populated unless the configuration parameter statistics_level is set to TYPICAL or ALL.

Rollback Contention Ratio

- Metrics

- Troubleshooting

Rollback segments are used by the Oracle RDBMS to hold data needed to rollback (or undo) any changes made through inserts, updates, or deletes to various Oracle objects. They also allow Oracle to have read consistency for long running queries. Rollback segments are used for recovery purposes and they play a role during exports of database information.

In a heavy transaction processing environment, rollback segments are accessed continuously and therefore are subject to contention problems. The rollback contention ratio helps identify contention occurring on the system relating to rollbacks.

> **TIP:** Click this statistic to drill down to the Rollback Waits tab of the Contention Detail view.

Metrics

Overall, if the rollback segment wait ratio approaches 1% or more, you should consider creating more rollback segments. You should also think about creating a specialized, larger, rollback segment to be used by long running transactions. Doing so alleviates dynamic rollback extensions and cuts down heavily on ORA-01555 snapshot too old errors.

Troubleshooting

Begin by creating new rollback segments and altering them to be online for use. Then monitor the overall contention ratio to see if it begins to drop.

Users Blocked

- Metrics
- Troubleshooting

On a small system, a single blocking user has the potential to stop work for nearly all other processes. On larger systems, this can cause major headaches. Although Oracle supports unlimited row-level locking, blocking lock situations do occur. User processes holding exclusive locks and not releasing them via a proper COMMIT frequency, generally cause most blocks.

> **TIP:** Click this statistic to drill down to the Blocking Locks tab of the Locks view.

Metrics

You should investigate any blocking lock statistic indicator above zero to prevent a mushrooming situation.

Troubleshooting

You can quickly remedy a blocking lock situation by issuing a KILL against the offending process. This eliminates the user's stranglehold on the accessed objects and lets other user processes complete. Tools like Performance Center make it easier to discover the blocked lock situation, the tricky part is preventing the blocking lock situation in the first place.

The culprit of blocking lock scenarios is usually the application design, or the SQL used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. Most DBAs who have had to face Oracle Forms applications have suffered through the dreaded SELECT…FOR UPDATE statements that place unnecessary restrictive locks on nearly every read operation, know all too well that good coding practice is important. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible, something not always easy to do.

Users Waiting (General)

User connections that are waiting on a system generally occur for two reasons:

- A process waits because a requested resource is not available.
- A process waits for Oracle to perform a prerequisite task for its given operation.

Idle waits (processes waiting because they have no work) should not worry you. The two wait causes mentioned above are worth your time and investigation.

NOTE: This statistic displays on both the Contention performance category view and the Users performance category view.

TIP: Click this statistic to drill down to the Session Waits tab of the Contention Detail view.

Metrics

To determine the actual wait causes user connections are experiencing, drill down from the global count of users waiting into the actual system and user wait details of your target database. This lets you locate the exact causes of currently experienced waits.

Troubleshooting

If you find a problem, drill down into wait details to determine whether the waits are resource related.

Sessions Waiting for Latches

User connections that are waiting on a system generally occur for two reasons:

1. A process waits because a requested resource is not available.
2. A process waits for Oracle to perform a prerequisite task for its given operation.

Sessions Waiting for Latches displays the number of user processes specifically waiting for latches.

TIP: Click this statistic to drill down to the Latch Waits tab of the Contention Detail view.

Metrics

If you find nonzero values for this statistic, you should access the drill-down views to discover the exact latches that are making users wait.

Sessions in Buffer Busy Waits

User connections that are waiting on a system generally occur for two reasons:

- A process waits because a requested resource is not available.
- A process waits for Oracle to perform a prerequisite task for its given operation.

Sessions involved in buffer busy waits indicates contention for data blocks in the buffer cache.

TIP: Click this statistic to drill down to the Buffer Busy Waits tab of the Contention Detail view.

Metrics

If you find nonzero values for this statistic, you should access the drill-down views to discover the exact latches that are making users wait.

Prolonged Lock Waits

This statistic shows the number of sessions that have been blocked for over 300 seconds.

On a small system, a single blocking user has the potential to stop work for nearly all other processes. On larger systems, this can cause major headaches. Although Oracle supports unlimited row-level locking, blocking lock situations do occur. User processes holding exclusive locks and not releasing them via a proper COMMIT frequency generally cause most blocks. Click this statistic to drill down to the Blocking Locks tab of the Locks view.

Metrics

You should investigate any indicator above zero to prevent a mushrooming situation.

Troubleshooting

You can quickly remedy a blocking lock situation by issuing a KILL against the offending process. This eliminates the user's stranglehold on the accessed objects and lets other user processes complete. Performance Center make it easier to discover the blocked lock situation, but the tricky part is preventing the blocking lock situation in the first place.

The culprit of blocking lock scenarios is usually the application design, or the SQL used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible, something not always easy to do.

Redo Log Space Requests

The Oracle RDBMS is able to manage recovery by recording all changes made to a database through the use of redo log files. Oracle writes modifications made to a database to the redo log files, which you can archive to another medium for disaster recovery. The background process that performs these operations is Oracle's Log Writer (LGWR). There is a buffer area in Oracle's System Global Area (SGA) that is used to reduce redo log file I/O, whose size, or lack thereof, can affect performance in a busy system. Sometimes a user process must wait for space in this redo log buffer. Oracle uses the log buffer area to cache redo entries prior to writing them to disk, and if the buffer is not large enough for the redo entry load, waits can occur.

TIP: Click this statistic to drill down to the LGWR/DBWR Detail tab of the I/O Detail view.

Metrics

The two main numbers to watch are:

1 Redo log space requests

2 Redo log wait time

If either statistic strays too far from zero, increase the log_buffer parameter and add more memory to the redo log buffer.

Troubleshooting

If you find a problem, do the following:

1 Edit the Init.ora file for the database.

2 Increase the amount of log_buffer to a higher value (take care not to over-allocate; ensure enough free memory exists on the server before increasing the value).

3 Cycle the Oracle server when possible to allow the new value to take effect.

4 Monitor the new value to see if performance improves.

When adjusting the log_buffer parameter, make sure that you make the amount a multiple of the block size. Otherwise, on database startup, Oracle returns an error stating that you have entered an invalid amount for the redo log buffer.

NOTE: Oracle silently increases the log_buffer parameter if you make it smaller than its default size for a given platform.

Redo Log Space Wait Time

The Oracle RDBMS is able to manage recovery by recording all changes made to a database through the use of redo log files. Oracle writes modifications made to a database to the redo log files, which you can archive to another medium for disaster recovery. The background process that performs these operations is Oracle's Log Writer (LGWR). There is a buffer area in Oracle's System Global Area (SGA) that is used to reduce redo log file I/O, whose size, or lack thereof, can affect performance in a busy system. Sometimes a user process must wait for space in the redo log buffer. Oracle uses the log buffer to cache redo entries prior to writing them to disk. If the buffer area is not large enough for the redo entry load, waits can occur. While waits can be an important indicator of contention, a better measure is the amount of wait time that your users experience.

TIP: Click this statistic to drill down to the LGWR/DBWR Detail tab of the I/O Detail view.

Metrics

The two main numbers to watch are:

1 Redo log space requests

2 Redo log wait time

If either statistic strays too far from zero, increase the log_buffer parameter and add more memory to the redo log buffer.

Troubleshooting

If you find a problem, do the following:

1. Edit the Init.ora file for the database.
2. Increase the amount of log_buffer to a higher value (take care not to over-allocate; ensure enough free memory exists on the server before increasing the value).
3. Cycle the Oracle server when possible to allow the new value to take effect.
4. Monitor the new value to see if performance improves.

When adjusting the log_buffer parameter, make sure you make the amount a multiple of the block size. Otherwise, on database startup, Oracle returns an error stating that you have entered an invalid amount for the redo log buffer.

NOTE: If you make the log_buffer parameter smaller than its default size for a given platform, Oracle silently increases it.

Parallel Query Busy Ratio

Oracle's parallel query feature, when used properly, allows large increases in performance for many SQL and utility operations. Parallel operations can be introduced through SQL hints, specified in an object's DDL, or used in command line arguments for utility programs like SQL*Loader. To effectively service parallel query requests, you must ensure that enough query servers exist in the database instance. The parallel query busy ratio is an indicator of how busy all the servers are on the database in question.

TIP: Click this statistic to drill down to the Parallel Query tab of the Contention Detail view.

Metrics

If the parallel query busy ratio approaches 80-90%, you should think about:

- Adding more query servers to the database.
- Examining parallel requests to ensure they are being used in an efficient and necessary manner.

Troubleshooting

If you find a problem, do the following:

1. Edit the Init.ora file for the database.
2. Increase the amount of parallel_max_servers to a higher value.
3. Cycle the Oracle server when possible to allow the new value to take effect.
4. Monitor the new value to see if performance improves.

You can also investigate the use of the parallel_automatic_tuning parameter in Oracle 8.1 or later.

Used Query Slaves

Oracle's parallel query feature, when used properly, allows large increases in performance for many SQL and utility operations. Parallel operations can be introduced through SQL hints, specified in an object's DDL, or used in command line arguments for utility programs like SQL*Loader. To effectively service parallel query requests, you must ensure that enough query servers exist in the database instance. The used query slave statistic displays how many query servers are actively performing work in a database.

> **TIP:** Click this statistic to drill down to the Parallel Query tab of the Contention Detail view.

Metrics

If the parallel query busy ratio approaches 80-90%, or you see all the available slaves being used, you should think about:

- Adding more query servers to the database.
- Examining parallel requests to ensure they are being used in an efficient and necessary manner.

Troubleshooting

If you find a problem, do the following:

1. Edit the Init.ora file for the database.
2. Increase the amount of parallel_max_servers to a higher value.
3. Cycle the Oracle server when possible to allow the new value to take effect.
4. Monitor new value to see if performance improves.

You can also investigate the use of the parallel_automatic_tuning parameter in Oracle 8.1 or later.

Max Query Slaves

Oracle's parallel query feature, when used properly, allows large increases in performance for many SQL and utility operations. Parallel operations can be introduced through SQL hints, specified in an object's DDL, or used in command line arguments for utility programs like SQL*Loader. To effectively service parallel query requests, you must ensure that enough query servers exist in the database instance. The max query slave statistic displays the maximum number of query slaves allowed on the system as dictated by the parallel_max_servers parameter.

> **TIP:** Click this statistic to drill down to the Parallel Query tab of the Contention Detail view.

Metrics

If the parallel query busy ratio approaches 80-90%, or you see all the available slaves being used, you should think about:

- Adding more query servers to the database.
- Examining parallel requests to ensure they are being used in an efficient and necessary manner.

Troubleshooting

If you find a problem, do the following:

1. Edit the Init.ora file for the database.
2. Increase the amount of parallel_max_servers to a higher value.

3 Cycle the Oracle server when possible to allow the new value to take effect.

4 Monitor the new value to see if performance improves.

Free List Waits

- Metrics

Free lists are lists of Oracle data blocks that contain free space for an Oracle object. Every table has at least one free list. You can use Free lists to locate free blocks of space when a request is made of a table for the insertion of a row. Free list contention can reduce the performance of applications when many processes are involved in the insertion of data to the same table.

TIP: Click this statistic to drill down to the System Waits tab of the Contention Detail view.

Metrics

If free list contention approaches 1%, you should add additional free lists to the most dynamic database objects through the use of the STORAGE parameter.

Enqueue Waits

- Metrics
- Troubleshooting

An enqueue is an advanced locking device that lets multiple database processes share certain resources. Enqueue waits typically occur when sessions wait to be granted a requested lock. Sometimes these locks are internal Oracle locks while other times they could be locks for rows of data in a table.

NOTE: Enqueues are issued implicitly by Oracle.

TIP: Click this statistic to drill down to the System Waits tab of the Contention Detail view.

Metrics

You should investigate any enqueue waits that read consistently above one or more (delta statistics).

Troubleshooting

Removing contention for enqueues is usually an application design issue. Many enqueue waits are either contention for certain rows in the database or the result of database-initiated lock escalation. You should examine the use of indexes to make sure all referencing foreign keys are indexes and that your SQL does not tarry over rows in the database during modification operations. If it does, you should tune your SQL.

Enqueue waits can also be the result of space management tasks (such as objects extending) and disk sorts (mainly in tablespaces that do not make use of the TEMPORARY tablespace parameter).

Contention Detail View

The following tabbed pages are available on the Contention Detail view:

- Buffer Busy Waits
- Latch Detail
- Latch Waits
- Parallel Query
- Rollback Waits
- Session Waits
- System Waits

Buffer Busy Waits Tab

- Metrics

Buffer busy waits occur when a process needs to access a data block in the buffer cache, but cannot, because it is being used by another process. Therefore, it must wait. The Buffer Busy Waits tab of the Contention Detail view shows how long users are waiting for buffers in the buffer cache. The table below describes the information available on the Buffer Busy Waits tab of the Contention Detail view:

Column	Description
Username	The user account being used by the session.
SID	The unique Oracle identifier for the session.
OSUser	The operating system ID of the user session.
Machine	The client machine name used by the session.
Program	The program being executed against the Oracle database by the session.
Total Waits	The total number of buffer busy waits for the session.
Time Waited	The total amount of time waited for the event, in hundredths of a second.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Buffer busy waits normally center around contention for rollback segments, too small an INITRANS setting for tables, or insufficient free lists for tables. The remedy for each situation would be increasing the number of rollback segments, or altering tables to have larger settings for INITRANS to allow for more transactions per data block, and more free lists.

Latch Detail Tab

- Metrics

Protecting the many memory structures in Oracle's SGA are latches. They ensure that one and only one process at a time can run or modify any memory structure at the same instant. Much more restrictive than locks (which at least allow for some collective user interaction), latches have no queuing mechanism so either you get it or you do not and are forced to continually retry.

The Latch Detail tab of the Contention Detail view presents a detailed view of latch activity. The table below describes the information available on the Latch Detail tab of the Contention Detail view:

Column	Description
Name	The name of the latch.
Gets	The number of times the latch was requested by a process.
Misses	The number of failed attempts to acquire the latch on the first attempt.
Spins	The number of times a failed first attempt acquired the latch on a spin.
Immediate Gets	The total number of nowait requests for a latch.
Immediate Misses	The total number of failed nowait attempts to acquire the latch on the first attempt.
Sleeps	The total number of requests that "paused" while waiting for a latch.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Common indicators of latch contention are a latch miss ratio (which records willing-to-wait mode latch requests) and latch immediate miss ratio (which records no-wait mode latch requests). These statistics reflect the overall health of how often latch requests were made and satisfied without waiting. If either of these exceeds 1%, latch contention can be causing system slowdowns.

The table below describes two latch contention situations that you can recognize and quickly fix:

Situation	Remedy
Cache Buffer Chain Latch	This latch is responsible for protecting paths to database block buffers in the buffer cache. Very high I/O loads tend to cause contention for this latch. You can alleviate contention somewhat by adding more buffers to the cache (through the db_block_buffers parameter) or by adding more LRU latch chain latches with the db_block_lru_latches parameter.
Library Cache Latches	Protects cached SQL statements in the library cache area of the Oracle shared pool. Contention for this latch is the usual result of using literals for SQL statements instead of using bind variables.

NOTE: Other routine latch contention problems included the redo allocation and redo copy latches, but these have pretty much been made obsolete in Oracle 8.1.5 or later.

Latch Waits Tab

- Metrics

Protecting the many memory structures in Oracle's SGA are latches. They ensure that one and only one process at a time can run or modify any memory structure at the same instant. Much more restrictive than locks (which at least allow for some collective user interaction), latches have no queuing mechanism so either you get it or you do not and are forced to continually retry. The Latch Waits tab of the Contention Detail view shows detailed information about current latch wait situations. The table below describes the information available on the Latch Waits tab of the Contention Detail view:

Column	Description
Username	The user account being used by the session.
SID	The unique Oracle identifier for the session.
Latch Name	The name of the latch.
Parameter1 Text	The description of the first wait parameter.
Parameter1	The first wait/tuning parameter.
Parameter2 Text	The description of the second wait parameter.
Parameter2	The second wait/tuning parameter.
Seq #	The sequence number that uniquely identifies the wait. It is incremented for each wait.
Wait	The time waited, in hundredths of a second.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Common indicators of latch contention are a latch miss ratio (which records willing-to-wait mode latch requests) and latch immediate miss ratio (which records no-wait mode latch requests). These statistics reflect the overall health of how often latch requests were made and satisfied without waiting. By using latch information on this view (contained in the parameter1 and parameter2 boxes) you can locate the exact object experiencing contention.

Parallel Query Tab

- Metrics

Oracle's parallel query option offers great potential in pursuit of faster response times. The ability to split a request into multiple segments that are processed in parallel can greatly speed up the race to a result, however it is important that Oracle be set up with the proper resources necessary to use parallel functions in an efficient manner. The Parallel Query tab of the Contention Detail view displays information relating to parallel query performance. The table below describes the information available on the Parallel Query tab of the Contention Detail view:

Column	Description
Parallel Query Statistic	The name of the statistic affecting parallel query performance.
Value	The statistical value.

Metrics

Seeing a server's busy statistic that nears or matches the maximum values allotted for parallel query slaves is a sure sign that more query slaves need to be added to the system. Also, seeing a server's highwater value that matches the value for the PARALLEL_MAX_SERVERS parameter can indicate a need to raise the number of query slaves given to Oracle.

Rollback Waits Tab

- Metrics

In systems with heavy data modification loads, the rollback segments can become quite a hot spot. Contention for rollback segments can occur between competing database transactions and cause a performance slowdown.

The Rollback Waits tab of the Contention Detail view displays all the rollbacks available on the system and their wait statistics. The table below describes the information available on the Rollback Waits tab of the Contention Detail view:

Column	Description
Name	The name of the rollback segment.
Gets	The number of header gets (the segment has been used).
Waits	The number of header waits.
WaitRatio	The individual contention ratio for the rollback segment.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Overall, if the global rollback contention ratio approaches 1% or more, you should consider creating more rollback segments. Seeing many individual rollback segments with wait ratios above 1% could also be a cause for concern.

You should also think about creating a specialized, larger, rollback segment for use by long running transactions. Doing so alleviates dynamic rollback extensions and cuts down dramatically on ORA-01555 snapshot-too-old errors. For a database under heavy DML loads, consider creating two rollback tablespaces on separate drives to minimize disk contention among the rollback segments.

Session Waits Tab

- Metrics

Session contention is merely a subset of contention that is viewable at the global database level. Often, it takes analysis at the session level to pinpoint the exact source of contention that is occurring globally. Therefore, you need to become accustomed to viewing contention statistics and waits at the user process level.

When monitoring waits with respect to user sessions, there are two areas of interest:

1. What HAS the user session been waiting on?
2. What IS the user session waiting on?

Oracle records both sets of wait statistics for you. In reviewing previous waits for a session, you can see what types of things have caused a bottleneck in the session. The table below describes the information available on the Sessions Waits tab of the Contention Detail view:

Column	Description
User	The user account being used by the session.
SID	The unique Oracle identifier for the session.
Wait Cause	The name of the wait cause
Program	The program being executed against Oracle by the user session.
Time (sec)	The number of seconds that process has been waiting.
Wait State	The value of the wait state. WAITING indicates that the session is waiting. WAITED UNKNOWN TIME indicates that the duration of last wait is unknown. WAITED SHORT TIME indicates that the last wait was less than 1/100th of a second. WAITED KNOWN TIME indicates that the wait is equal to the time of the last wait.
Object Owner	The name of the object owner.
Object Name	The name of the object.

NOTE: This information is available on both the Session Waits tab of the Users Detail view and the Session Waits tab of the Contention Detail view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

To view the wait times for sessions and overall system wait events, you must set the TIMED_STATISTICS parameter to TRUE for your Oracle databases. You can either set this parameter in your Init.ora file or alter the system dynamically with the ALTER SYSTEM SET TIMED_STATISTICS=TRUE command.

Some waits can be ignored like SQL*Net more data from client and SQL*Net message from client. Others like enqueue waits can indicate a lock contention problem. Waits, like db file scattered reads, can indicate sessions involved in long table scan operations.

System Waits Tab

- Metrics

Waits on a system generally occur for three reasons:

1. A process waits because it has no work to do.
2. A process waits because a requested resource is not available.
3. A process waits for Oracle to perform a prerequisite task for its given operation.

Idle waits (processes waiting because they have no work) should not worry you at all, however the other two wait causes are the ones worth your time and investigation. From a global database level, there are many different types of waits and sources of contention. The System Waits tab of the Contention Detail view presents all the various system waits that have occurred on the system since startup. The table below describes the information available on the System Waits tab of the Contention Detail view:

Column	Description
Wait Event	The name of the wait event.
Total Waits	The total number of waits for the event.
Total Timeouts	The total number of timeouts for the event.
Time Waited	The total amount of time waited for the event in hundredths of a second.
Average Wait Time	The average amount of time waited for the event in hundredths of a second.
Percent Total	The total percentage of waits this event makes up relative to all waits on the system.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Which waits should concern you and which waits can you ignore (for the most part)? Common wait events to ignore include:

- SQL*Net more data from client
- SQL*Net message from client
- client message
- rdbms ipc message
- pipe get
- pmon timer
- smon timer
- Null event

If you see many enqueue waits, this normally indicates either contention for certain rows in the database, or the result of database-initiated lock escalation. You should examine the use of indexes to make sure all referencing foreign keys are indexes and that your SQL is tuned to not tarry over rows in the database during modification operations. Enqueue waits can also be the result of space management tasks (such as objects extending) and disk sorts (mainly in tablespaces that do not make use of the TEMPORARY tablespace parameter).

Buffer busy waits normally center around contention for rollback segments, too small an INITRANS setting for tables, or insufficient free lists for tables. The remedy for each situation would be increasing the number of rollback segments, or altering tables to have larger settings for INITRANS to allow for more transactions per data block, and more free lists.

An interesting wait event is the db file scattered read event. This event is indicative of table scans occurring on the system. Large numbers of them can indicate heavy scan activity and the need to revisit your indexing/physical design.

Network Statistics - Oracle

The Network performance category view displays the following vital network statistics:

- Bytes Received from Client

Bytes Sent to Client

- Metrics

The total number of bytes sent over the network to all Oracle client machines from the database since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Out of the ordinary numbers should raise alarms because possible network saturation could occur.

Bytes Received from Client

- Metrics

The total number of bytes sent over the network to the Oracle database from all client machines since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Out of the ordinary numbers should raise alarms because possible network saturation could occur.

Roundtrips to/from Client

- Metrics

The total number of Oracle network messages sent to and received from the client.

Metrics

All users connected to a database via a client/server connection generates network traffic, so seeing decent volumes of network traffic is normal. Out of the ordinary numbers should raise alarms because possible network saturation could occur.

Bytes Sent to DBLink

- Metrics

The total number of bytes sent by all client machines over the network to any database link since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Out of the ordinary numbers should raise alarms because possible network saturation could occur.

> **NOTE:** Database link traffic could experience longer response times depending on the distributed nature and set up of the involved databases.

Bytes Received from DBLink

- Metrics

The total number of bytes received by all client machines over the network from any database link since the last refresh.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Out of the ordinary numbers should raise alarms because possible network saturation could occur.

> **NOTE:** Database link traffic could experience longer response times depending on the distributed nature and setup of the involved databases.

Roundtrips to/from DBLink

- Metrics

The number of Oracle network messages sent over and received from a database link.

Metrics

All users connected to a database via a client/server connection generate network traffic, so seeing decent volumes of network traffic is normal. Out of the ordinary numbers should raise alarms because possible network saturation could occur. Also note that database link traffic could experience longer response times depending on the distributed nature and setup of the involved databases.

Network Contention

- Metrics
- Troubleshooting

Sometimes after sending a request to the database, a user experiences a slowdown in getting a request back from the Oracle server. These delays are usually negligible with respect to the actual amount of time that the user process actually waits. Examining the amount of network-related waits in light of all waits experienced by the database indicates the percentage of waits that the network is contributing to the overall contention scenario.

Metrics

The main thing to do with respect to your database and the network is to minimize the amount of data that travels over it. This means ensuring that only the amount of data your users truly need is being returned to their query tools or front-end application.

Troubleshooting

Eliminating unnecessary large volumes of rows being sent over the wire to clients from the Oracle database is one place to start, as overall traffic can be reduced.

MTS Response Activity

In the past, database professionals avoided Oracle's Multi-threaded server option unless they had a server that was low on horsepower. Instead, most people used the direct option of each user having his or her own SQL*Net thread. This was (and probably still is) the fastest way for Oracle networking to perform its job.

The MTS Response Activity area displays information regarding the amount of waits in the MTS response queues managed by Oracle's shared dispatcher processes.

> **NOTE:** These statistics are only available for Oracle servers with the Multi-threaded server option.

Metrics

When the number of waits for MTS responses begins to climb, adding more shared dispatchers can increase performance.

Troubleshooting

You can add more dispatchers to Oracle both by editing the Init.ora file and changing the mts_dispatchers and mts_max_dispatchers parameters to higher amounts (and then cycling Oracle) or dynamically changing the mts_dispatchers parameter through use of the ALTER SYSTEM command.

MTS Request Activity

In the past, database professionals avoided Oracle's Multi-threaded server option unless they had a server that was low on horsepower. Instead, most people used the direct option of each user having his or her own SQL*Net thread. This was (and probably still is) the fastest way for Oracle networking to perform its job.

The MTS Request Activity area displays information regarding the amount of waits in the MTS request queues managed by Oracle's shared dispatcher processes.

> **NOTE:** These statistics are only available for Oracle servers with the Multi-threaded server option.

Metrics

When the number of waits for MTS requests begins to climb, performance can be increased by adding more shared dispatchers.

Troubleshooting

You can add more dispatchers to Oracle by both editing the Init.ora file and changing the mts_dispatchers and mts_max_dispatchers parameters to higher amounts (and then cycling Oracle) or dynamically changing the mts_dispatchers parameter through use of the ALTER SYSTEM command.

Users Page Statistics

The Users home page includes the following sections:

- Active Connections
- Current Locks
- Current Transactions
- Inactive Connections
- Leading Sessions - CPU
- Leading Sessions - I/O
- Leading Sessions - Memory
- Max Connections
- Max Open Locks
- Max Sessions
- Max Transactions
- Open Cursors
- Sessions Involved in Disk Sorts
- Total Sessions
- Users Blocked
- Users Waiting

Related Topics

Users Detail

Home View Statistics

I/O Page Statistics

Memory Page Statistics

Objects Page Statistics

OS Page Statistics

Space Page Statistics

Top SQL Page Statistics

Active Connections

The Active Connections statistic is the total number of active and open threads currently reported in the database as well as the number of processes actively performing work.

Metrics

None.

> **TIP:** Click this statistic to drill down to the Sessions Overview tab of the Users Detail view.

Current Locks and Max Open Locks

- Metrics
- Troubleshooting

This set of statistics reflects the total number of DML locks currently held on the database as well as the maximum number of DML locks allowed on the system at one time.

For more information, see Current Locks.

> **TIP:** Double-click these statistics to drill down to the All Locks tab of the Locks view.

Metrics

Seeing the current locks on the database approach 80-90% of the maximum limit allowed indicates that you should increase the Init.ora parameter dml_locks.

Troubleshooting

If the total number of locks on the database approaches the dml_locks limit, then:

1. Ensure that users are efficiently issuing COMMITs in transactions to release held locks before editing the Init.ora file.
2. Edit the Init.ora file for the database.
3. Increase the amount of dml_locks to a higher value.
4. Cycle the Oracle server when possible to allow the new value to take effect.

Current Transactions and Max Transactions

- Metrics
- Troubleshooting

This set of statistics reflects the total number of open transactions currently on the database as well as the maximum number of transactions allowed on the system at one time.

> **TIP:** Double-click these statistics to drill down to the Transaction Detail tab of the Users Detail view.

Metrics

Seeing the current transactions on the database approach 80-90% of the maximum allowed limit allowed indicates that you should increase the Init.ora parameter transactions.

Troubleshooting

If the total number of transactions on the database approaches the transactions limit, you should:

1. Edit the Init.ora file for the database.
2. Increase the amount of transactions to a higher value.
3. Cycle the Oracle server when possible to allow the new value to take effect.

Total Connections and Max Connections

This set of statistics reflects the total number of open sessions on the database (both active and inactive) as well as the maximum number of sessions allowed on the system at one time.

> **TIP:** Double-click Total Connections or Max Connections to drill down to the Sessions Overview tab of the Users Detail view.

Metrics

Seeing the open connections on the database approach 80-90% of the maximum process limit allowed indicates that you should increase the process limit imposed on the database.

Troubleshooting

If the total number of connections on the database approaches the processes limit, then:

1. Ensure that users are efficiently logging off and on the database before editing the Init.ora file.
2. Edit the Init.ora file for the database.
3. Increase the amount of processes to a higher value.
4. Cycle the Oracle server when possible to allow the new value to take effect.

Inactive Connections

The Inactive Connections statistic is the total number of threads logged on to the database that are currently idle.

> **TIP:** Click this statistic to drill down to the Sessions Overview tab of the Users Detail view.

Metrics

A large number of inactive users could indicate user sessions that have mistakenly been left logged on. Because each user thread consumes a portion of memory on the Oracle server, to reduce resource usage, you should sever any session that does not need a connection.

Troubleshooting

Drill down into the Sessions Overview tab of the Users Detail view and check sessions that have many seconds idle and/or have been logged on for very long periods of time (as indicated by the logon time column). After verifying that a session is no longer necessary, you can KILL it.

Open Cursors

- Metrics
- Troubleshooting

Open Cursors is the total number of all SQL open cursors that exist on the system. In some cases, Oracle cached cursors that have been open by PL/SQL procedures can be kept open for certain lengths of time, even though the actual activity has ceased.

TIP: Click this statistic to drill down to the Open Cursors tab of the Users Detail view.

Metrics

You should monitor sessions to make sure that they do not approach the Open Cursor limit (specified in the Init.ora file). The parameter, open_cursors, limits how many open cursors (context areas) a session can have open at one time.

Troubleshooting

If the total number of open cursors approaches the open_cursors limit on Oracle8i or earlier, then:

- Ensure that user processes are efficiently using cursors before editing the Init.ora file.
- Edit the Init.ora file for the database.
- Increase the amount of open_cursors to a higher value.
- Cycle the Oracle server when possible to allow the new value to take effect.

If the total number of open cursors approaches the open_cursors limit on Oracle9i or later then:

- Change the open_cursors parameter to FORCE by using the ALTER SYSTEM SET open_cursors=< new value > command.

Users Waiting

- Metrics
- Troubleshooting

User connections that are waiting on a system generally occur for two reasons:

1. A process waits because a requested resource is not available.
2. A process waits for Oracle to perform a prerequisite task for its given operation.

Idle waits (processes waiting because they have no work) are not usually a concern. However the two wait causes mentioned above are the ones worth your time and investigation.

NOTE: This statistic displays on both the Contention performance category view and the Users performance category view.

TIP: Click this statistic to drill down to the Session Wait Detail tab of the Users Detail view.

Metrics

To determine the actual wait causes currently experienced by user connections, you should drill down from the global count of users waiting, into the actual system and user wait details of a database.

Troubleshooting

If you find a problem, drill down into wait details to determine whether the waits are resource-related.

Users Blocked

A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches even on large systems. Although Oracle supports unlimited row-level locking, blocking lock situations do crop up. User processes holding exclusive locks and not releasing them via a proper COMMIT generally cause most blocks.

> **TIP:** Click this statistic to drill down to the Blocking Locks tab of the Locks view.

Metrics

You should immediately investigate any indicator above zero for a blocking lock statistic before the situation has a chance to grow out of control.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied. You can issue a KILL against the offending process, which eliminates the user's stranglehold on the objects they were accessing. Other user processes then nearly almost always complete in an instant. Performance Center makes it easier to discover the blocked lock situation, but the trick is to prevent the blocking lock situation in the first place.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. Most DBAs who have had to face Oracle Forms applications have suffered through the dreaded SELECT … FOR UPDATE statements that place unnecessary restrictive locks on nearly every read operation, and know all too well that good coding practice is important. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

Sessions Involved in Disk Sorts

Oracle's SGA is not the only memory structure used by Oracle for database work. One of the other memory areas used by Oracle8i and earlier for normal activity is an area set aside for sort actions. When a sort operation occurs, Oracle attempts to perform the sort in a memory space that exists at the operating system level. If the sort is too large to be contained within this space, it continues the sort on disk - specifically, in the user's assigned TEMPORARY TABLESPACE. Oracle records the overall number of sorts that are satisfied in memory as well as those that end up being finalized on disk. Using these numbers, you can calculate the percentage of memory sorts vs. disk sorts and get a feel for how fast your sort activity is being resolved. Obviously, memory sorts completes many times faster than sorts forced to use physical I/O to accomplish the task at hand.

Oracle9i or later now has the option of running automatic PGA memory management. Oracle has introduced a new Oracle parameter called pga_aggregate_target. When the pga_aggregate_target parameter is set and you are using dedicated Oracle connections, Oracle ignores all of the PGA parameters in the Oracle file, including sort_area_size, hash_area_size and sort_area_retained_size. Oracle recommends that the value of pga_aggregate_target be set to the amount of remaining memory (less a 10% overhead for other server tasks) on a server after the instance has been started.

TIP: Click this statistic to drill down to the Disk Sort Detail tab of the Users Detail view.

Metrics

Consistently seeing non-zero numbers for this statistic (as well as low values observed for the memory sort ratio) can indicate excessive disk sort activity. If you are on Oracle8i or earlier, increase the parameters devoted to memory sorts - sort_area_size and sort_area_retained_size.

If you are using Oracle9i or later, investigate the use of pga_aggregate_target. Once the pga_aggregate_target has been set, Oracle automatically manages PGA memory allocation, based upon the individual needs of each Oracle connection. Oracle9i or later allows the pga_aggregate_target parameter to be modified at the instance level with the alter system command, thereby lets you dynamically adjust the total RAM region available to Oracle9i.

Oracle9i also introduces a new parameter called workarea_size_policy. When this parameter is set to automatic, all Oracle connections benefit from the shared PGA memory. When workarea_size_policy is set to manual, connections allocate memory according to the values for the sort_area_size parameter. Under the automatic mode, Oracle tries to maximize the number of work areas that are using optimal memory and uses one-pass memory for the others.

Troubleshooting

If you find a problem, do the following:

- Edit the Init.ora or SPFILE file for the database.
- Increase the amount of sort_area_size to a higher value (take care not to not over-allocate; ensure enough free memory exists on server before increasing value). Realize that EVERY user receives this amount for sorting).
- Cycle the Oracle server when possible to allow the new value to take effect.
- Monitor new value to see if performance improves.

In addition to increasing the amount of memory devoted to sorting, you should also locate inefficient SQL that causes needless sorts. For example, in an SQL query (to eliminate duplicate rows) UNION ALL does not cause a sort whereas UNION does. People frequently code DISTINCT inappropriately (especially people transferring from Microsoft Access, which uses DISTINCT for most SELECT queries).

There are times you simply cannot stop sort activity. When this happens, you should try to keep it in memory whenever possible. However, large data warehousing systems frequently exhaust RAM sort allotments, so if disk sorts must occur, ensure three things:

- Your user's TEMPORARY TABLESPACE assignment is not the SYSTEM tablespace, which is the default assignment. In Oracle9i or later, you can specify a default tablespace other than SYSTEM for every user account that is created.
- The TEMPORARY TABLESPACE assigned to your users is placed on a fast disk.
- The TEMPORARY TABLESPACE has the tablespace parameter TEMPORARY assigned to it, which allows sort activity to be performed in a more efficient manner.

Leading Sessions - CPU

- Metrics

It is not uncommon for one or two users to cause the majority of runtime problems that plague a database. The problem could be a runaway process, an untuned batch procedure, or other user-initiated operation. Often, user connections can get out of hand with CPU use, and extreme cases have caused headaches at both the database and operating system levels.

The leading CPU session's display allows you to see who runs the leading sessions in your system with respect to CPU usage.

Metrics

Finding one or two users who use the majority of the CPU can indicate runaway or improper processes. By drilling down into the CPU activity of all users, you can quickly see if this is the case.

Leading Sessions - Memory

It is not uncommon for one or two users to cause the majority of runtime problems that plague a database. The problems could include a runaway process, an untuned batch procedure, or some other user-initiated operation. Frequently, user connections can get out of hand with memory consumption, with extreme cases causing headaches at both the database and operating system levels (ORA-4030 errors).

NOTE: This statistic displays on both the Users performance category view and the Memory performance category view.

TIP: Click this statistic to drill down to the Leading Sessions tab of the Memory Detail view.

Metrics

If your database server does not have an overabundance of memory, you should periodically check to see who your heavy memory users are and the total percentage of memory they take. If you see one or two users who have more than 5-15% of the total memory usage, you should investigate the sessions further to see what activities they are performing.

Troubleshooting

The Leading Sessions - Memory statistic lets you find the users with the greatest allocations of overall memory. You can drill down into the details to discover the memory components that make up each user's session. The table below describes areas where you should pay particular attention to the amounts of memory usage:

Area	Description
PGA	The Program Global Area is a private memory area devoted to housing the global variables and data structures for a single Oracle process.
Memory Sorts	Contains a count of how many memory sorts a session has performed.
UGA	The User Global Area contains session-specific information regarding open cursors, state information for packages, database link information, and more.

TIP: When using Oracle's Multi-threaded Server (MTS), the UGA can be moved up into the SGA.

Leading Sessions - I/O

When a system undergoes heavy I/O activity, you find that all the user connections are contributing somewhat equally to the overall load. Often however, one or two user connections are responsible for 75% or more of the I/O activity. It can be that a large batch load or another typical process is running that is perfectly okay for your system or there can be a runaway process or rogue connection that you need to track down and possibly eliminate.

The Leading I/O Session statistic displays, with respect to I/O, the leading sessions in your system.

NOTE: This statistic displays on both the Users performance category view and the I/O performance category view.

TIP: Click this statistic to drill down to the Leading Sessions tab of the I/O Detail view.

Metrics

Finding one or more users who are consuming more than 75% of the total I/O load can indicate a runaway or improper process. By drilling down into the I/O activity of all users, you can quickly see if this is the case.

Users Detail

The following tabbed pages are available on the Users Detail page:

- Disk Sort Detail
- Open Cursors
- Session Waits
- Sessions Overview
- System Tablespace
- Transaction Detail

Disk Sort Detail Tab

- Metrics

Something that can degrade a user's or overall database performance is disk sort activity. When a sort operation occurs, Oracle attempts to perform the sort in a memory space, assigned by the DBA, which exists at the operating system level. If the sort is too large to be contained within this space, it continues the sort on disk - specifically, in the user's assigned TEMPORARY TABLESPACE.

The Disk Sort Detail tab of the Users Detail view displays information relating to active disk sorts on a system. The table below describes the information available on the Disk Sort Detail tab of the Users Detail view:

Column	Description
User	The user account a session is using.
SID	The unique Oracle identifier for the session.
Tablespace	The tablespace being used for the disk sort.
Contents	Whether the tablespace has been set to PERMANENT or TEMPORARY.
Extents	The number of extents involved in the sort.
Blocks	The number of blocks involved in the sort.
Bytes	The total number of bytes used in the sort.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Techniques to include in your overall performance strategy are those that relate to minimizing the amount of sort activity overall and specifically sort activity that takes place on disk. A good place to start is by understanding what things cause sorts in the first place. A list of sort-related commands and SQL-related options include:

- CREATE INDEX, ALTER INDEX…REBUILD
- DISTINCT
- ORDER BY
- GROUP BY
- UNION
- INTERSECT
- MINUS
- IN, NOT IN
- Certain unindexed joins
- Certain correlated subqueries

All of these SQL commands have the potential to create a sort. As a DBA, you probably won't know which query sorts perform entirely in memory and which ones go to disk.

There are times you simply cannot stop disk sort activity (such as in data warehousing environments). That being the case, you should ensure the following are true in your database:

- Your users' TEMPORARY TABLESPACE assignment is not the SYSTEM tablespace, which is the default assignment.
- The TEMPORARY TABLESPACE assigned to your users is placed on a fast disk.
- The TEMPORARY TABLESPACE has the tablespace parameter TEMPORARY assigned to it, which allows sort activity to be performed in a more efficient manner.

If your overall memory sort ratio falls below 90%, increase the parameters devoted to memory sorts - sort_area_size and sort_area_retained_size. Keep in mind that Individual users might have the ability to alter their own sessions and increase their sort_area_size assignments. As a DBA, restrict users that have the ALTER SESSION privilege.

In addition to increasing the amount of memory devoted to sorting, you should also locate inefficient SQL that causes needless sorts. For example, in an SQL query (to eliminate duplicate rows) UNION ALL does not cause a sort whereas UNION does. People frequently code DISTINCT inappropriately (especially people transferring from Microsoft Access, which uses DISTINCT for most SELECT queries).

Open Cursors Tab

- Metrics

Programs opening cursors consume Oracle resources and have the potential to degrade performance, especially if their SQL code is inefficient. The Open Cursors tab of the Users Detail view allows you to quickly spot user accounts that have many cursors opened as well as the actual performance statistics for each opened cursor. The first grid displays a summary of the open cursors. The second grid displays the details of open cursors.

The table below describes the information available in the Open Cursor Summary grid on the Open Cursors tab of the Users Detail view:

Column	Description
User	The user account owning the cursors.
SID	The unique Oracle identifier for the session.
Open Cursors	The number of open cursors for the session.

The table below describes the information available in the Open Cursor Detail grid on the Open Cursors tab of the Users Detail view:

Column	Description
SID	The unique Oracle identifier for the session.
SQL Text	The SQL text being used for the cursor.
Disk reads	The number of physical I/O reads.
Buffer gets	The number of buffer gets.
Rows	The number of rows returned for the statement.
Parse Calls	The total of all parse calls to all the child cursors under this parent.
Sharable Memory	The sum of all sharable memory of all child cursors under this parent, in bytes.
Persistent Memory	The sum of all persistent memory of all child cursors under this parent, in bytes.
Runtime Memory	The sum of all ephemeral frame sizes of all children.
Sorts	The sum of sorts that were done for all the children of the parent code.
Loaded Versions	The number of children that are present in the cache and have their context heap loaded.
Loads	The number of times the object was loaded or reloaded.
Executions	The number of times the code was executed since being brought into the shared pool.
First Load Time	The creation timestamp of the parent code.

TIP: To configure a grid to display row numbers, use the Options Editor.

Metrics

The Init.ora parameter OPEN_CURSORS controls the maximum number of open cursors (context areas) a session can have at one time. Seeing individual sessions approaching this limit should concern you. To avoid cursor allocation errors, set it to a very high number. The cursor limit for the Init.ora file was 255, but this limit has been increased in Oracle8.

With respect to individual cursor performance statistics, be on the lookout for cursors with abnormally high physical I/O counts (which can indicate inefficient SQL or indexing schemes), and cursors with high load counts (which can indicate an undersized library cache).

Session Waits Tab

Session contention is merely a subset of contention that is viewable at the global database level. Often, it takes analysis at the session level to pinpoint the exact source of contention that is occurring globally. Therefore, you need to become accustomed to viewing contention statistics and waits at the user process level.

When monitoring waits with respect to user sessions, there are two areas of interest:

- What HAS the user session been waiting on?
- What IS the user session waiting on?

Oracle records both sets of wait statistics for you. In reviewing previous waits for a session, you can see what types of things have caused a bottleneck in the session. The table below describes the information available on the Sessions Waits tab of the Users Detail view:

Column	Description
User	The user account being used by the session.
SID	The unique Oracle identifier for the session.
Wait Cause	The name of the wait cause
Program	The program being executed against Oracle by the user session.
Time (sec)	The number of seconds that process has been waiting.
Wait State	The value of the wait state. WAITING indicates that the session is waiting. WAITED UNKNOWN TIME indicates that the duration of last wait is unknown. WAITED SHORT TIME indicates that the last wait was less than 1/100th of a second. WAITED KNOWN TIME indicates that the wait is equal to the time of the last wait.
Object Owner	The name of the object owner.
Object Name	The name of the object.

NOTE: This information is available on both the Session Waits tab of the Contention Detail view and the Session Waits tab of the Users Detail view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

To view the wait times for sessions and overall system wait events, you must set the TIMED_STATISTICS parameter to TRUE for your Oracle databases. You can either set this parameter in your Init.ora file or alter the system dynamically with the ALTER SYSTEM SET TIMED_STATISTICS=TRUE command.

Some waits can be ignored like SQL*Net more data from client and SQL*Net message from client. Others like enqueue waits can be indicative of a lock contention problem. Waits, like db file scattered reads, can indicate sessions involved in long table scan operations.

Sessions Overview Tab

- Metrics

You can get more detail into dips in the overall buffer cache hit ratio by examining user I/O activity and users' individual cache hit ratios. Frequently you can pinpoint one or more accounts responsible for reducing performance and examine their SQL and other statistics to fix the situation.

The table below describes the information available on the Sessions Overview tab of the Users Detail view:

Column	Description
User	The logon name the session is using.
SID	The unique Oracle identifier for the session.
Serial #	The serial number assigned to the session.
Status	The status of the session, ACTIVE or INACTIVE.
Machine	The name of the client machine that the session is using.
O/S ID	The unique operating system identifier for the target session.
Logon Time	The date/timestamp of when the session first logged on.
Blocked	Whether the session is blocked from doing work by another session.
Seconds Idle	The number of seconds since the last time the session actively performed work.
Hit Ratio	The percentage of times the session obtained data from memory vs. physical I/O activity. The maximum value is 100%.
Program	The program being executed against Oracle by the user session.

NOTE: This information is available on both the Sessions Overview tab of the Memory Detail view and the Sessions Overview tab of the Users Detail view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Observing a poor hit ratio for one or more threads can help you pinpoint sessions that are dragging overall database performance down. Obtaining the current SQL for those sessions could yield information into the exact nature of the problem. Typically, you should investigate hit ratios under 85%. Another critical piece of information is sessions blocked. You should investigate these statistics using the Performance Center Locks view. Two other interesting pieces of information include the logon time and seconds idle. Sessions with high idle times and logon time stamps that are many hours or days old could be candidates for removal from the system.

System Tablespace Tab

- Metrics

Of all your tablespaces, you want to avoid fragmentation problems in your SYSTEM tablespace the most, as this is the major hotbed tablespace for Oracle activities. The easiest way to avoid this is to not allow any user (even the default DBA ID's SYS and SYSTEM) to have access to it. There are three ways to do this:

1 Ensure no user has a DEFAULT or TEMPORARY tablespace assignment of SYSTEM.

2 Ensure no user has a quota set for SYSTEM.

3 Ensure no user has been granted the UNLIMITED TABLESPACE privilege.

TIP: The System Tablespace tab of the Users Detail view identifies users that have privileges on the SYSTEM tablespace.

Metrics

Seeing any user with SYSTEM tablespace privileges should be cause for alarm. To rectify the situation, you can do one of the following:

- Use the ALTER command to change a user's DEFAULT or TEMPORARY tablespace assignment from SYSTEM to a user-defined tablespace.
- Remove any system tablespace quotas assigned to users.
- Revoke the UNLIMITED TABLESPACE privilege from any identified user.

Transaction Detail Tab

- Metrics

Long running transactions have the potential to consume large amounts of resources and cause lock contention headaches. The Transaction Detail tab of the Users Detail view provides a snapshot of transactions in progress. The table below describes the information available on the Transaction Detail tab of the Users Detail view:

Column	Description
User	The user account used by the session.
SID	The unique Oracle identifier for the session.
Rollback Segment	The name of the rollback segment.
Machine	The client machine name the session is using.
Program	The program being executed against Oracle by the session.
Status	The current status of the transaction.
Start Time	The beginning timestamp for the transaction.
Logical	The total logical I/O for the transaction.
Physical	The total physical I/O for the transaction.
Gets	The total consistent gets used by the transaction.
Changes	The total consistent changes produced by the transaction.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Seeing a long running, persistent transaction might indicate a session caught in a loop or other coding mess. Watching a transaction sit idle could clue you into a lock contention problem that exists.

Session Details

The following tabbed pages are available on the Session Detail view for Oracle:

- Session Contention
- Session I/O
- Session Memory
- Session Network
- Session Objects
- Session SQL
- Session Statistics

Session Memory Tab for Oracle

- Metrics

The Session Memory tab of the Session Detail view presents the statistics surrounding a session's memory usage. The table below describes the information available on the Session Memory tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the memory related statistic.
Value	The cumulative value for the memory statistic.
Cache Hit Ratio	The percentage of data obtained from memory access vs. physical I/O.

Metrics

Sessions with abnormally high memory usage can affect overall performance at the server level, as this memory (PGA and UGA) is allocated outside of the Oracle SGA (unless the multi-threaded server option is being used). If cache hit ratios at the session level are lower than 85 percent, data access can be inefficient.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session I/O Tab for Oracle

- Metrics

The Session I/O tab of the Session Detail view presents the statistics surrounding a session's I/O activity. The table below describes the information available on the Session I/O tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the I/O related statistic.
Value	The cumulative value for the I/O statistic.

Metrics

Seeing high values for physical reads and writes can indicate an inefficient session. Large numbers of physical reads can imply a session with too many large table scans or inefficient SQL operations. Large numbers of physical writes can be okay for sessions inputting large volumes of data into the database. Or, they could also indicate a session involved in heavy disk sort activity.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session Contention Tab for Oracle

- Metrics

The Session Contention tab of the Session Detail view presents information relating to resources on which the current session is waiting. The first grid displays user waits. The second grid displays user locks. The table below describes the information available in the User Waits grid on the Session Contention tab of the Session Detail view for Oracle:

Column	Description
Wait Cause	The wait event being experienced by the session.
Program	The program the session is executing against Oracle.
Seconds	The number of seconds the session has spent in the wait.
State	The status of the wait event (WAITING, etc.)

The table below describes the information available in the User Locks grid on the Session I/O tab of the Session Detail view for Oracle:

Column	Description
User	The user account being used by the session.
Terminal	The client machine name used by the session.
SID	The unique Oracle identifier for the session.
Serial #	The serial number for the session.
Table	The object locked by the session.
Lock Mode	The lock mode used by the session.
Title	The lock request issued by the session.

Metrics

You can ignore some waits, like the SQL*Net more data from client and SQL*Net message from client. Others, like enqueue waits, can be indicative of a lock contention problem. Waits, like db file scattered reads, can indicate sessions involved in long table scan operations.

Locks that are held for unusually long periods require further investigation. The application logic can be inefficient or the program is not issuing COMMITs frequently enough. The culprit of blocking-lock scenarios is usually the application design, or the SQL used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the correct SQL to get the job done is an art. Most DBAs who have had to face Oracle Forms applications have suffered through the dreaded SELECT…FOR UPDATE statements. These place unnecessary restrictive locks on nearly every read operation. The key to avoiding lock contention is to process user transactions as quickly and efficiently as possible.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session Objects Tab for Oracle

- Metrics

The Session Objects tab of the Session Detail view presents information regarding the objects in use by a session. The table below describes the information available in the Objects Accessed grid on the Session Objects tab of the Session Detail view for Oracle:

Column	Description
Owner	The owner of the object.
Type	The type of object (TABLE, VIEW, etc.).
Object	The name of the object.

The Rollback Segments Accessed grid displays the names of rollback segments currently being used by the session.

Metrics

Once you have an idea of which objects your users access most often, you can refine some processes to facilitate access to them. You can use the Oracle 8 concept of the KEEP buffer cache to force Oracle to hold often-referenced data for data objects. The KEEP buffer cache is ideal for holding small look-up tables. If you have an earlier version of Oracle or do not want to split up your current buffer cache, you can use the CACHE table attribute to encourage Oracle to keep data blocks of CACHE'd tables at the most recently used end of the LRU buffer cache chain.

If you consistently see a session with active rollbacks, it can indicate locks are being held for long durations. It can also indicate that a session is using code without frequent enough COMMIT points.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session Network Tab for Oracle

- Metrics

The Session Network tab of the Session Detail view presents information about requests being sent to and from the database with respect to the current session. The table below describes the information available on the Session Network tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the SQL*Net related statistic.
Value	The cumulative value of the SQL*Net related statistic.

Metrics

None.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session SQL Tab for Oracle

- Metrics

The Session SQL tab of the Session Detail view presents information relating to any current SQL issued by the session as well as any SQL previously issued by the session.

Metrics

To determine access paths, you should export and run through an EXPLAIN PLAN session any SQL that you suspect of inefficient access.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session Statistics Tab for Oracle

- Metrics

The Session Statistics tab of the Session Detail view presents information relating to all recorded performance and miscellaneous statistics for the current session. The table below describes the information available on the Session Statistics tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the session related statistic.
Value	The cumulative value of the session related statistic.

Metrics

None.

> **NOTE:** To configure the grid to show/hide row numbers, use the Options Editor.

Other Views and Statistics

In addition to the Home view, Enterprise view, and the performance category views, Performance Center offers many other views. The tables below lists, by database platform, the other views available in Performance Center:

View	Oracle	SQL Server	Sybase	DB2
Alert Log	x	x	x	x
Archive	x			
Configuration Parameters		x	x	x
Health Index	x	x	x	x
Hot Objects	x			
Instance Parameters	x			
Lock	x	x	x	x
Operating System	x	x	x	x
Session Detail	x	x	x	x
SQL Server Logs		x		
Top Sessions	x	x	x	x
Top SQL	x	x		x
Trends	x	x	x	x

Archive View

- Metrics

To allow for point-in-time recovery, Oracle writes copies of redo log information to disk. When a database is running in archive log mode, a DBA, with proper backup techniques in place, can recover nicely from a database error and roll forward to almost any point in time as long as the proper archive logs are in place.

The I/O needed to write these archive logs is handled by Oracle's ARCH process. The Archive view allows archive files written by the ARCH process to be viewed by user-specified time frames. The table below describes the information available on the Archive view:

Column	Description
Date/Time	The timestamp of the archive log (when the log was written).
Title	The actual archive log file name and path.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Numerous archive files can be written to disk if there is heavy redo log activity. Batch jobs have the potential to move very fast; sometimes so fast that the online redo logs wrap back around before they have a chance to be archived. Messages indicating this has happened show up in the Oracle alert log. If this happens frequently, you should think about increasing the size of the online redo log files, or increasing the number of redo logs in general.

Seeing archive files written at a rate of more than one every 30-60 minutes can indicate the redo size is too small (or there is an above-average data modification load).

If you do not want to lose an archive file that can be needed for recovery and you are using Oracle 8 or later, you can take advantage of the feature where you can write archive files to more than one destination on disk. This feature also allows multiple ARCH processes to be invoked. Investigate using the Init.ora parameters log_archive_dest_n and log_archive_max_processes.

Always remember one thing with respect to archive files and running Oracle in archive log mode: Running out of archive file space on the server can halt all activity in a database. Make sure you have ample free space available on your archive drives. And, you should also implement a purge procedure for older archives in conjunction with your backup routine.

Health Index View

- Metrics

Performance Center's global and category-specific health indexes are fast and efficient indicators you can use to determine if a database is experiencing problems. They also locate the most problematic area(s). With the Health Index view, you can scan individual category indexes simultaneously and see, over time, where the problem areas reside.

Metrics

Generically speaking, you should investigate any index that falls below 90. Temporary dips in a health index graph should not be a cause for concern unless the dips form a pattern and occur on a predictable and continuous basis.

Hot Objects

The following tabbed pages are available on the Hot Objects view:

- Hot Tables
- Hot Code

NOTE: The Hot Objects view is for Oracle datasources.

Hot Tables

- Metrics

Certain objects in an Oracle database are accessed more than others. These objects can become a source of contention given certain conditions. The Hot Tables tab of the Hot Objects view identifies tables that are being frequently accessed through various SQL statements. The table below describes the information that Performance Center displays on the Hot Tables tab of the Hot Objects view:

Column	Description
Table Owner	The owner of the table.
Table Name	The name of the table.
Command Issued	The SQL statement command issued against the table.
Executions	The number of SQL executions the object has experienced.

Column	Description
Disk Reads	The number of estimated disk reads from the object.
Buffer Gets	The number of estimated logical I/O's from the object.
Rows Processed	The number of estimated rows processed from the object.

Metrics

DML activity against tables can cause contention for space management objects like free lists. Oracle9i and above provides automatic segment management, which can remove problems with free lists and the like.

Hot Code

- Metrics

Certain objects in an Oracle database are accessed more than others. Data objects can become a source of contention given certain conditions, while code objects rarely cause contention issues. The Hot Code tab of the Hot Objects view identifies code objects (procedure, functions, etc.) that are being frequently accessed through various SQL statements. The table below describes the information that Performance Center displays on the Hot Code tab of the Hot Objects view:

Column	Description
Object Owner	The owner of the object.
Object Name	The name of the object.
Object Type	The type of object (package, etc.)
# of Executions	The number of estimated executions for the object.
Loads	The number of times the object was loaded into the shared pool.
Locks	The number of locks the object has experienced.
Pins	The number of times the object was pinned in the shared pool.

Metrics

Often referenced code objects should be pinned in the shared pool using the Oracle DBMS_SHARED_POOL package. Objects with many executions and loads should be considered candidates for pinning.

Lock View

The following tabbed pages are available on the Lock view:

- All Locks
- All User Locks (Oracle only)
- Blocking Locks
- Locks View for DB2

All Locks Tab

The information on the All Locks tab of the Locks view depends on the target DBMS:

- Oracle
- SQL Server
- Sybase

All Locks Tab for Oracle

- Metrics

To modify database information or structures, a user session must obtain a lock on the object to perform its task. In addition to user locks, Oracle issues lock requests to carry out its internal duties. The All Locks tab of the Locks view displays information about all locks currently on a system. The table below describes the information available on the All Locks tab of the Locks view for Oracle:

Column	Description
SID	The session identifier of the session holding the lock.
User Name	The user account of the session holding the lock. NULL if it is a background process.
Lock Mode	The lock mode (EXCLUSIVE, SHARE, etc.)
Request Type	The type of lock requested by the session.
Object Name	The name of the object being locked.
Object Type	The type of object being locked (TABLE, etc.)
Lock Type	The type of lock (TRANSACTION, DML, etc.)
Lock ID 1	The lock identifier #1 (depends on type).
Lock ID 2	The lock identifier #2 (depends on type).

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Locks held for unusually long periods are candidates for further investigation. The application logic can be inefficient or the program may not be issuing COMMIT frequently enough.

All Locks Tab for SQL Server

- Metrics

To modify database information or structures, a user session must obtain a lock on the object to perform its task. In addition to user locks, SQL Server issues lock requests to carry out its internal duties. Performance Center displays information about all locks currently on a system on the All Locks tab of the Locks view and the Lock Detail grid on the Lock tab of the Contention Detail view.

The table below describes the information available on the tab and the grid:

Column	Description
SPID	The process ID of the process holding the lock.
Login	The login name of the process.
NT User	The operating system name of the process.
Database	The database where the locks are occurring.
Table Name	The name of the table involved in a lock. This will be NULL for non-table locks or table locks that take place in the tempdb database.
Ndx ID	The index ID involved in the lock.
Lock Type	The type of lock (database, table, row ID, etc.)
Lock Mode	The mode of the lock (shared, exclusive, etc.)
Lock Status	The status of the lock (waiting or granted).
Owner Type	Whether the lock came from a regular session or a transaction.
Program	The executable the process is using against the server.
BLK SPID	If nonzero, the process ID of the process blocking the requested lock. A value of zero indicates that the process is not blocked.
Wait Time	The time the process has waited for the lock, in milliseconds.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Command	The command the process is currently issuing.
NT Domain	The name of the Windows 2000/NT domain.

NOTE: The information in the Lock Detail grid is available in the Lock Detail grid on the Locks tab of the Contention Detail view and the All Locks tab of the Locks view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Locks held for unusually long periods are candidates for further investigation. The application logic may be inefficient or the program may not be issuing COMMIT frequently enough.

All Locks Tab for Sybase

- Metrics

To modify database information or structures, a user session must obtain a lock on the object to perform its task. In addition to user locks, Sybase issues lock requests to carry out its internal duties. The All Locks tab of the Locks view displays information about all locks currently on a system. The table below describes the information available on the All Locks tab of the Locks view for Sybase:

Column	Description
PID	The process ID.
User Name	The user name assigned to the process.
Database	The database in which the process is running.
Lock Type	The type of lock (database, table, row ID, etc.)
Object Name	The name of the object being locked.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Lock Page	The page number where the lock is currently applied.
Lock Class	The name of the cursor the lock is associated with (if any).
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Command	The command the process is currently issuing.
CPU Time	The CPU time accumulated for the current command.
Physical I/O	The current cumulative number of reads and writes issued by the process.
Mem Usage	The memory accumulated for the current command.
FID	The process ID of the worker process' parent.
Transaction	The name of any transaction.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Locks held for unusually long periods are candidates for further investigation. The application logic may be inefficient or the program may not be issuing COMMIT frequently enough.

All User Locks Tab

- Metrics

To modify database information or structures, a user session must obtain a lock on the object to perform its task. The All User Locks tab of the Locks view displays information about all user locks currently on a system. The table below describes the information available on the All User Locks tab of the Locks view:

Column	Description
User Name	The user account that holds the lock.
Terminal	The machine name of the client session.
SID	The unique Oracle identifier for the session.
Serial #	The serial number of the lock.
Table	The name of the object being locked.
Lock Mode	The lock mode (EXCLUSIVE, SHARE, etc.)
Request	The type of lock requested by the session.

NOTE: The All User Locks tab of the Locks view is only available for Oracle datasources.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Locks held for unusually long periods are candidates for further investigation. The application logic may be inefficient or the program may not be issuing COMMITs frequently enough.

Blocking Locks Tab

The information on the Blocking Locks tab of the Locks view depends on the target DBMS.

- Oracle
- SQL Server
- Sybase

Blocking Locks Tab for Oracle

- Metrics

Blocking-lock situations can make the database appear frozen, rivalling only a stuck archive in effect. A single blocking user has the potential to stop work for nearly all other processes on a small system, or can cause major headaches on large systems. Although Oracle supports unlimited row-level locking, blocking-lock situations do occur - sometimes frequently.

The Blocking Locks tab of the Locks view contains information relating to user accounts that are currently blocked and the sessions that are blocking them. The table below describes the information available on the Blocking Locks tab of the Locks view for Oracle:

Column	Description
Blocked SID	The session ID of the session waiting for the lock.
Blocked User	The user account of the session waiting for the lock.
Wait Time (sec)	The current wait time for the session, in seconds.
Blocking SID	The session ID of the session holding the offending lock.
Blocking User	The user account of the session holding the offending lock.
Lock Type	The type of lock (TRANSACTION, DML, etc.)
Lock Mode	The lock mode (EXCLUSIVE, SHARE, etc.)
Request Type	The type of lock being requested by the session.
Locked Object	The name of the object being locked.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

When a blocking lock is discovered, the DBA can quickly remedy the situation by issuing a KILL against the offending process. This eliminates the user's stranglehold on the objects the user was accessing. Other user processes then usually complete in an instant. Tools like Performance Center make it easier to discover the blocking-lock situation.

The culprit of blocking-lock scenarios is often the application design, or the SQL used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. Most DBAs who have had to face Oracle Forms applications have suffered through the dreaded SELECT…FOR UPDATE statements that place unnecessary restrictive locks on nearly every read operation. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

Data warehouses, whose data is mostly read, can benefit from tablespaces set in read-only mode. Read-only mode signals to the other databases that exclusive locks need not be used for the data contained within the tablespace. This is especially helpful in Oracle Parallel Server environments and drastically reduces ping activity.

Blocking Locks Tab for SQL Server

- Metrics

Performance Center displays information about all blocking locks currently on a system on the Blocking Locks tab of the Locks view and the Blocking Lock Detail grid on the Blocking Lock tab of the Contention Detail view.

The table below describes the information available on the tab and the grid:

Column	Description
SPID	The process ID of the process holding the lock.
Login	The login name of the process.
NT User	The operating system name of the process.
Database	The database where the locks are occurring.
Table Name	The name of the table involved in a lock. This will be NULL for non-table locks or table locks that take place in the tempdb database.
Ndx ID	The index ID involved in the lock.
Lock Type	The type of lock (database, table, row ID, etc.).
Lock Mode	The mode of the lock (shared, exclusive, etc.).
Lock Status	The status of the lock (waiting or granted).
Owner Type	Whether the lock came from a regular session or a transaction.
Program	The executable the process is using against the server.
BLK SPID	If nonzero, the process ID of the process blocking the requested lock. A value of zero indicates that the process is not blocked.
Wait Time	The time the process has waited for the lock, in milliseconds.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Command	The command the process is currently issuing.
NT Domain	The name of the Windows 2000/NT domain.

NOTE: The information in the Blocking Lock Detail grid is available in the Blocking Lock Detail grid on the Blocking Locks tab of the Contention view and the Blocking Locks tab of the Locks view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

When a blocking lock is discovered, the DBA can quickly remedy the situation by issuing a KILL against the offending process. This eliminates the user's stranglehold on the objects the user was accessing. Other user processes then usually complete in an instant. Tools like Performance Center make it easier to discover the blocking-lock situation.

Blocking Locks Tab for Sybase

- Metrics

The Blocking Locks tab of the Locks view contains information relating to user accounts that are currently blocked and the sessions that are blocking them. The table below describes the information available on the Blocking Locks tab of the Locks view for Sybase:

Column	Description
Holding PID	The process ID that owns the blocking lock.
Holding User	The user account of the session holding the offending lock.
Waiting PID	The session PID of the session waiting for the lock.
Waiting User	The user account of the session waiting for the lock.
Database	The database in which the process is running.
Object Name	The table on which the lock is being held.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Lock Type	The type of lock being applied.
Time Blocked	The time that a process has been waiting for the lock, in seconds.
Lock Page	The page number where the lock is currently applied.
Lock Class	The name of the cursor the lock is associated with (if any).
Holding Host	The name of the host computer with the blocking lock.
Waiting Host	The name of the host computer waiting for the lock.
Holding Program	The program the process is running that has the lock.
Waiting Program	The program the process is running that is waiting for the lock.
Holding Command	The command being issued by the process holding the lock.
Waiting Command	The command being issued by the process waiting for the lock.
CPU Time	The CPU time accumulated for the current command.
Physical I/O	The physical I/O accumulated for the current command.
Mem Usage	The memory accumulated for the current command.
Holding FID	The process ID of the worker process' parent that has the lock.
Waiting FID	The process ID of the worker process' parent that is waiting for the lock.
Transaction	The name of the associated transaction (if any).

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

When a blocking lock is discovered, the DBA can quickly remedy the situation by issuing a KILL against the offending process. This eliminates the user's stranglehold on the objects the user was accessing. Other user processes then usually complete in an instant. Tools like Performance Center make it easier to discover the blocking-lock situation.

Locks View for DB2

The Locks view displays all processes that are currently holding locks on an IBM DB2 UDB database. The following sections of this view are available to display lock information:

- Applications
- Locks Held Tab

- Locks Waiting Tab
- Unit of Work Tab

Applications

This section lists the following lock information for all applications:

Agent ID – The application handle of the agent holding a lock for which this application is waiting.

Auth ID – The authorization ID of the user who invoked the application that is being monitored.

OS User ID – The authorization ID used to access the operating system.

Client PID – The process ID of the client application that made the connection to the database.

Application – Name of the application executable.

Status – The lock's status (waiting or granted).

Locks Held – The number of locks on the lock being held.

Locks Waiting – Indicates the number of agents waiting on a lock

Lock Wait Time (ms) – The current amount of wait time for the process, in milliseconds.

Timeouts – The number of times that a request to lock an object timed out without being granted.

Deadlocks – Processes that cannot proceed because they are waiting on a set of resources held by each other or held by other processes.

Locks Held Tab

This tab displays all the locks held by the selected application in the Applications list. The following data is available:

Lock Mode – The type of lock being held.

Object Type – The type of object against which the application holds a lock (for object-lock-level information), or the type of object for which the application is waiting to obtain a lock (for application-level and deadlock-level information).

Table Schema – Schema of the table that the lock is on.

Table Name – Name of the table that the lock is on. This element is only set if Object Type indicates Table.

Tablespace – The name of the table space against which the lock is held.

Lock Status – The lock's status (waiting or granted).

Escalation – Indicates whether a lock request was made as part of a lock escalation.

Locks Waiting Tab

This tab displays all the locks waiting by the selected application in the Applications list. The following data is available:

Agent ID – The application handle of the agent holding a lock for which this application is waiting.

Application ID – The application ID of the application that is holding a lock on the object that this application is waiting to obtain.

Lock Mode – The type of lock being held.

Mode Requested – The lock mode being requested by the application.

Object Type – The type of object against which the application holds a lock.

Table Schema – Schema of the table that the lock is on.

Table Name – Name of the table that the lock is on. This element is only set if Object Type indicates Table.

Tablespace – The name of the table space against which the lock is held.

Wait Start Time – The date and time that this application started waiting to obtain a lock on the object that is currently locked by another application

Escalation – Indicates whether a lock request was made as part of a lock escalation.

Unit of Work Tab

This tab displays the SQL statement text for the selected application. The statement is available if the selected application is in lock wait status or is the thread blocking other applications. This enables you to easily identify what SQL statements are causing lock wait conditions and to help diagnose deadlock scenarios.

Click **Explain SQL** to view an explain plan for the statement.

Operating System View

The Operating System view displays vital operating system statistics on the following tabbed pages:

- Summary
- CPU
- Processes
- I/O
- Memory
- Space
- Network

To use integrated security, or gather certain operating system statistics, there are two main things to know:

1 You must enable the Performance Center Server for operating system monitoring. To enable it, you must select the Enable operating system monitoring option on the Machine tab of the Datasource Properties dialog.

2 You must supply the credentials necessary to view these statistics. To be able to view performance counters on a remote computer, Microsoft requires specific permissions on the remote computer that you want to monitor.

   Because the Performance Center Server collects data using the registry, monitoring a remote computer requires the use of the Remote Registry Service. If the service stops due to failure, the system restarts it automatically only once. Therefore, if the service stops again, you must manually restart it. You can change this default behavior by modifying the properties for Remote Registry Service. To access service properties, see Services under Services and Applications in Computer Management or see Administrative Tools. You can also check the Event Viewer's Application and System Logs for events that might have stopped the service.

   Remote data collection also requires access to specific registry subkeys and system files. To provide remote access to the registry to collect data on remote systems, Microsoft requires that users have a minimum of Read access to the Winreg subkey in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServers. By default, members of the Administrators group have Full Control access and members of the Backup Operators group have Read access. Microsoft also requires that users have Read access to the registry subkey that stores counter names and descriptions used by System Monitor, HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib*LanguageID*, where *LanguageID* is the numeric code for the spoken language for the operating system installation. (For English, the subkey is Perflib\009.) By default, Microsoft gives Full Control access to the System account and members of the Administrators and Creator Owners groups. Therefore, a local user on a server who is not logged in as an administrator cannot see performance counters.

   In addition, users might also require read access to the files that supply counter names and descriptions to the registry, Perfc*.dat and Perfh*.dat. (where the asterisk is a wildcard character representing the specific language code; for English, these are Perfc009.dat and Perfh009.dat.) If these files reside on an NTFS volume, to have access to them, the access control lists (ACLs) on these files must specify that the user has such access. By default, members of the Administrators and Interactive groups have sufficient access.

OS Page Statistics

In many scenarios, an optimally tuned database may not perform well because there are constraints imposed by the system where the database is running. These constraints may include processes competing with the database sever for resources (CPU, I/O, or Memory), a slow CPU, insufficient or slow I/O devices, and insufficient memory. In the OS Statistics page of Performance Center you can examine operating system metrics for the following platforms:

- AIX
- HP-UX

  NOTE: To view processor info and swap disk info on an HP-UX box, you need to login as ROOT in the OS login.

- Linux
- Solaris
- Unix
- Windows XP, 2000, and NT

Summary Tab

The OS Summary tab displays the following statistics to communicate the general overall performance levels of the operating system:

- Disk Time
- Load Average
- Processor Time
- Paged Memory Used (Windows)
- Swap Memory Used (AIX, HP-UX, Linux, Solaris, Unix)
- Average Disk Queue
- Network Output Queue (Windows)
- Network Queue (Solaris)
- Page Faults/Sec
- Processor Queue
- Processor Speed
- Processor
- Available Paged Memory (Windows)
- Available Physical Memory
- Available Swap Memory (AIX, HP-UX, Linux, Solaris, Unix)
- Total Paged Memory (Windows)
- Total Physical Memory
- Total Swap Memory (AIX, HP-UX, Linux, Solaris, Unix)
- Free Disk Space
- Total Disk Space
- Used Disk Space
- Number of Logins
- Number of Processes
- Number of Processors
- Top CPU Process
- Top I/O Process
- Top Memory Process

Processor Time

The Processor Time statistic indicates the percentage of time the processor is working. This counter is a primary indicator of processor activity.

Metrics

If your computer seems to be running sluggishly, this statistic could be displaying a high percentage.

Troubleshooting

Upgrade to a processor with a larger L2 cache, a faster processor, or install an additional processor.

Processor Speed

The Processor Speed statistic displays the speed of the active processor in MHz. The speed is approximate.

Processor

The Processor Statistic displays the type of processor currently in use, for example, GenuineIntel.

Disk Time

The Disk Time statistic is the percentage of elapsed time that the selected disk drive/device was busy servicing read or write requests.

Metrics

You should avoid consistently seeing values for this statistic greater then 90%.

Troubleshooting

Add more disk drives and partition the files among all of the drives.

Load Average

The Load Average statistic represents the system load averages over the last 1, 5, and 15 minutes.

Metrics

High load averages usually mean that the system is being used heavily and the response time is correspondingly slow.

Paged Memory Used

The Paged Memory Used statistic is the ratio of Commit Memory Bytes to the Commit Limit. Committed memory is where memory space has been reserved in the paging file if it needs to be written to disk. The commit limit is determined by the size of the paging file. As the paging file increases, so does the commit limit.

NOTE: This statistic is available for the Windows platform.

Metrics

This value displays the current percentage value only and not an average. If the percentage of paged memory used is above 90%, you may be running out of memory.

Troubleshooting

Increase the size of page file.

Number of Processors

This statistic displays the number of processors currently in use.

Swap Memory Used

The Swap Memory Used statistic is the percentage of swap space currently in use.

Metrics

If the percentage of swap memory used is above 90%, you may be running out of memory.

Troubleshooting

Increase the size of your swap files.

Average Disk Queue

The Average Disk Queue statistic is the average number of both read and write requests that were queued for the selected disk during the sample interval.

Metrics

This metric is useful in identifying I/O related bottlenecks. If the disk queue lengths for certain disks are consistently much higher than others, you may need to redistribute the load among available disks. If the disk queues lengths for all disks are consistently large, and you see a high amount of I/O activity, your disks may be inefficient.

Troubleshooting

Some things you can do if you have problems with this statistic include:

- Redistribute the data on the disk with the large average disk queue to other disks.
- Upgrade to faster disk(s).

Page Faults/Sec

The Page Faults/Sec statistic is the overall rate faulted pages are handled by the processor. It is measured in numbers of pages faulted per second. A page fault occurs when a process requires code or data that is not in its working set. This counter includes both hard faults and soft faults.

Metrics

This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

If the number of page faults remains consistently high, you can check with your Windows System Administrator for further investigation. Often, large numbers of page faults are not a problem so long as they are soft faults. However, hard faults, that require disk access, can cause delays.

Processor Queue

The Processor Queue Length statistic is the number of threads in the processor queue.

Metrics

Unlike the disk counters, this counter shows ready threads only, not threads that are running. There is a single queue for processor time even on computers with multiple processors. Therefore, if a computer has multiple processors, you need to divide this value by the number of processors servicing the workload. A sustained processor queue of less than 10 threads per processor is normally acceptable, dependent of the workload.

Troubleshooting

A sustained high value in the Processor Queue could indicate that a processor bottleneck has developed due to threads of a process requiring more process cycles than are available. If this is the case, you should look at installing a faster (or an additional) processor.

Network Output Queue/Network Queue

The Network Output Queue Length statistic is the number of threads in the processor queue.

NOTE: The name of this statistic depends on the platform of the operating system.

Metrics

Unlike the disk counters, this counter shows ready threads only, not threads that are running. There is a single queue for processor time even on computers with multiple processors. Therefore, if a computer has multiple processors, you need to divide this value by the number of processors servicing the workload. A sustained processor queue of less than 10 threads per processor is normally acceptable, dependent of the workload.

Troubleshooting

A sustained high value in the Processor Queue Length could indicate that a processor bottleneck has developed due to threads of a process requiring more process cycles than are available. If this is the case, you should look at installing a faster (or an additional) processor.

Available Physical Memory

The Available Physical Memory statistic represents the amount of RAM available to all processes.

Metrics

This counter displays the last observed value only and not an average. Use this value with the Total physical memory and paging metrics (Memory details page). If the available physical memory is very small compared to this value, and the paging activity is high, your system may be running low on memory.

Troubleshooting

Some things you can do if you have problems with this statistic include:

- Check the running processes to see if there are any memory leaks.
- Stop any services that are not required.
- Install additional RAM.

Available Paged Memory

The Available Paged Memory statistic shows the amount of virtual memory available for the processes.

NOTE: This statistic is available for the Windows platform.

Metrics

If the available virtual memory is less than 10% of the total virtual memory, your system may run out of memory.

Troubleshooting

Increase the size of page file.

Available Swap Memory

The Available Swap Memory statistic represents the amount of virtual memory available for the processes.

Metrics

If the available Available Swap Memory is less than 10% of the total Swap Memory, your system may run out of memory.

Troubleshooting

Increase the size of swap files.

Total Physical Memory

The Total Physical Memory statistic shows the amount of physical memory installed on your computer.

Metrics

This is an informational metric and displays the total amount installed on the machine. Use this value with the available physical memory and paging metrics (Memory details page). If the available physical memory is very small compared to this value, and the paging activity is high, your system may be running low on memory.

Total Paged Memory/Total Swap Memory

The Total Paged Memory statistic shows the maximum amount of virtual memory available to all processes.

NOTE: The name of this statistic depends on the platform of the operating system.

Metrics

It is recommended that this value is between 1.5 to 3 times the amount of RAM on the system.

Used Disk Space

The Used Disk Space statistic shows the amount of allocated space, in megabytes on all logical disk drives.

Troubleshooting

There are many things a DBA can do to ensure that a database does not encounter a space problem due to physical space limitations:

- If a database currently resides on a disk that has little free space, you can add more files to the database. Of course, you should add the new files to other physical hard disks that can accommodate a growing database.
- You should examine hard disks with shrinking disk space to see if you can relocate or delete files to allow more free space.

Total Disk Space

Total Disk Space displays the total allocated and unallocated space, in megabytes on all logical disk drives.

Troubleshooting

There are many things a DBA can do to ensure that a database does not encounter a space problem due to physical space limitations, here are two:

1. If a database currently resides on a disk that has little free space, you can add more files to the database. Of course, you should add the new files to other physical hard disks that can accommodate a growing database.
2. You should examine hard disks with shrinking disk space to see if you can relocate or delete files to allow more free space.

Free Disk Space

The Free Disk Space statistic shows the unallocated space, in megabytes on all logical disk drives.

Troubleshooting

There are many things a DBA can do to ensure that a database does not encounter a space problem due to physical space limitations:

- If a database currently resides on a disk that has little free space, you can add more files to the database. Of course, you should add the new files to other physical hard disks that can accommodate a growing database.
- You should examine hard disks with shrinking disk space to see if you can relocate or delete files to allow more free space.

Top Memory Process

Top Memory Process shows the current process that is consuming the most amount of memory. The information displayed is dependent on the platform of the operating system. Information displayed includes the name of the process, process ID, amount of memory consumed expressed in KB, amount of CPU expressed as a percentage, the amount of Major Page Faults, and the amount of I/O expressed in KB/sec.

Metrics

If you are running out of memory on the system, this is a quick way to identify the top memory user. If the displayed process is using a significant portion of the total memory, it could be causing the memory issues.

Processes Overview

The Processes Overview of the OS Summary includes the following sections:

- Top CPU Process
- Top I/O Process
- Top Memory Process

Top CPU Process

Top CPU Process shows the current process that is consuming the most amount of CPU. The information displayed is dependent on the platform of the operating system. Information displayed includes the name of the process, process ID, amount of memory consumed expressed in KB, amount of CPU expressed as a percentage, the amount of Major Page Faults, and the amount of I/O expressed in KB/sec.

Metrics

If the amount of CPU time used by this process is close to 100% and the CPU usage is very high, this process may be the bottleneck on the server.

Troubleshooting

Investigate the process further to see if it is in an inconsistent state. Also, look at minimum requirements for CPU speed for the process. You may need to upgrade your CPU.

Top I/O Process

The Top I/O Process statistic shows the current process that is consuming the most amount of CPU. The information displayed is dependent on the platform of the operating system. Information displayed includes the name of the process, process ID, amount of memory consumed expressed in KB, amount of CPU expressed as a percentage, the amount of Major Page Faults, and the amount of I/O expressed in KB/sec.

Number of Logins

This statistic displays the total number of logins on the server.

Number of Processes

This statistic displays the total number of processes on the server.

CPU Tab

The CPU tab of the OS Detail includes the following sections:

- Context Switches/Sec
- CPU Utilization
- Interrupts/Sec
- Processor Queue Length

CPU Utilization

The CPU Utilization section includes the following information:

- % Privileged Time
- % User Time

% Privileged Time

The % Privileged Time statistic is the percentage of elapsed time that the process threads spent executing code in privileged mode.

> **NOTE:** For Windows systems, when a Windows system service is called, the service will often run in privileged mode to gain access to system-private data. Such data is protected from access by threads executing in user mode. Calls to the system can be explicit or implicit, such as page faults or interrupts. These kernel commands, are considered privileged to keep the low-level commands executing and prevent a system freeze. Unlike some early operating systems, Windows uses process boundaries for subsystem protection in addition to the traditional protection of user and privileged modes. Some work done by Windows on behalf of the application might appear in other subsystem processes in addition to the privileged time in the process.

Metrics

The ideal range should be 0-40% (less than 40% indicates excessive system activity).

Troubleshooting

If your CPU consistently runs at less than 40% you may need to upgrade your system to include a faster processor(s).

% User Time

The % User Time statistic is the percentage of elapsed time the processor spends in the user mode. User mode is a restricted processing mode designed for applications, environment subsystems, and integral subsystems. The alternative, privileged mode, is designed for operating system components and allows direct access to hardware and all memory. The operating system switches application threads to privileged mode to access operating system services. This counter displays the average busy time as a percentage of the sample time.

Metrics

If the Privileged Time is high in conjunction with Physical Disk Reads, consider upgrading the disk I/O subsystem.

Interrupts/Sec

The Interrupts/Sec statistic is the average rate, in incidents per second, at which the processor received and serviced hardware interrupts. It does not include deferred procedure calls (DPCs), which are counted separately. This value is an indirect indicator of the activity of devices that generate interrupts, such as the system clock, the mouse, disk drivers, data communication lines, network interface cards, and other peripheral devices. These devices normally interrupt the processor when they have completed a task or require attention. Normal thread execution is suspended. The system clock typically interrupts the processor every ten milliseconds, creating a background of interrupt activity. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Metrics

The ideal rage should be 0-5000. A number greater then 5000 indicates possible excessive hardware interrupts; justification is dependent on device activity.

Context Switches/Sec

The Context Switches/Sec section shows the combined rate at which all processors on the computer are switched from one thread to another. Context switches occur when a running thread voluntarily relinquishes the processor, is preempted by a higher priority ready thread, or switches between user-mode and privileged (kernel) mode to use an Executive or subsystem service.

Metrics

The ideal range should be between 0-10,000. GA number greater then 10,000 may indicate too many threads contending for resources.

Processor Queue Length

The Processor Queue Length statistic is the number of threads in the processor queue. There is a single queue for processor time even on computers with multiple processors.

> **NOTE:** For Windows systems, unlike the disk counters, this counter shows ready threads only, not threads that are running.

Metrics

A sustained high value in the Processor Queue Length could indicate that a processor bottleneck has developed due to threads of a process requiring more process cycles than are available. If this is the case, you should look at installing a faster (or an additional) processor.

Processes Tab

The Processes tab of the OS Detail page succinctly communicates the general overall performance levels of processes. The columns available in this table depend on the platform of operating system. The table below describes the information available in the table on this tab:

Column	Description
Process	The name of the process.
User	The user of the process.
ID	The ID Process is the unique identifier of this process. ID Process numbers are reused, so they only identify a process for the lifetime of that process.
CPU	The CPU is the percentage of elapsed time that all of process threads used the processor to execution instructions.
User Mode	The User Mode is the percentage of elapsed time that the process threads spent executing code in user mode.
Memory WINDOWS ONLY	Memory is the current size, in bytes, of the virtual address space the process is using. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. Virtual space is finite, and the process can limit its ability to load libraries.
Memory (MB)	Memory is the current size, in bytes, of the virtual address space the process is using. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. Virtual space is finite, and the process can limit its ability to load libraries.
Memory	Memory is the percentage of the memory used of the total memory.
Active Memory	Active Memory is the amount of committed virtual memory, in bytes for this process. Active memory is the physical memory which has space reserved on the disk paging file(s). There can be one or more paging files on each physical drive. This counter displays the last observed value only; it is not an average.

Column	Description
I/O Data	The rate at which the process is reading and writing bytes in I/O operations. This counter counts all I/O activity generated by the process to include file, network and device I/Os.
Elapsed Time	The total elapsed time, in seconds, that this process has been running.
Thread Count	The number of threads currently active in this process. An instruction is the basic unit of execution in a processor, and a thread is the object that executes instructions. Every running process has at least one thread.
Handle Count	The total number of handles currently open by this process. This number is equal to the sum of the handles currently open by each thread in this process.
Priority	The current base priority of this process. Threads within a process can raise and lower their own base priority relative to the process' base priority.
Creating Proc ID	The Creating Process ID value is the Process ID of the process that created the process. The creating process may have terminated, so this value may no longer identify a running process.
Page Faults/Sec	Page Faults/Sec is the rate at which page faults by the threads executing in this process are occurring. A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory. This may not cause the page to be fetched from disk if it is on the standby list and hence already in main memory, or if it is in use by another process with whom the page is shared.
Page File	Page File is the current number of kilobytes that this process has used in the paging file(s). Paging files are used to store pages of memory used by the process that are not contained in other files. Paging files are shared by all processes, and the lack of space in paging files can prevent other processes from allocating memory.
Private	Private is the current size, in kilobytes, of memory that this process has allocated that cannot be shared with other processes.

I/O Tab

The table below describes the information available in this section:

Column	Description
Disk	The disk number assignment.
Reading (KB/s)	The amount of bytes read from the device.
Writing (KB/s)	The amount of bytes written to the device.
Disk Read Time	Disk Read Time is the percentage of elapsed time that the selected disk drive was busy servicing read requests.
Disk Write Time	Disk Write Time is the percentage of elapsed time that the selected disk drive was busy servicing write requests.
Disk Time	Disk Time is the percentage of elapsed time that the selected disk was busy servicing requests.
Avg. Read Queue	Avg. Disk Read Queue Length is the average number of read requests that were queued for the selected disk during the sample interval.
Avg. Write Queue	Avg. Disk Write Queue Length is the average number of write requests that were queued for the selected disk during the sample interval.
Disk Reads/Sec	Disk Reads/Sec is the rate of read operations on the disk.
Disk Writes/Sec	Disk Writes/Sec is the rate of write operations on the disk.

Memory Tab

The Memory tab of the OS Detail page includes the following sections:

- Cache Efficiency
- Free Physical
- Free Paged
- Paging Activity
- Page Faults
- Total Physical
- Total Paged

Paging Activity

The Paging Activity section includes the following statistics:

- Pages Input/Sec
- Pages Output/Sec

Pages Input/Sec

The Pages Input/Sec statistic is the number of pages read from disk to resolve hard page faults. Hard page faults occur when a process requires code or data that is not in its working set or elsewhere in physical memory, and must be retrieved from disk.

Metrics

This value was designed as a primary indicator of the kinds of faults that cause system-wide delays. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) and in non-cached mapped memory files. This counter counts numbers of pages, and can be compared to other counts of pages, such as Memory: Page Faults/sec, without conversion. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

Although it never hurts to have as much physical memory as your system can handle, there are some things you can check within your system to alleviate the memory bottleneck.

- Check to see if you have any drivers or protocols that are running but not being used. They use space in all memory pools even if they are idle.
- Check to see if you have additional space on your disk drive that you could use to expand the size of your page file. Normally, the bigger the initial size of your page file, the better, in performance terms.

Pages Output/Sec

The Pages Output/Sec statistic is the number of pages written to disk to free up space in physical memory. Pages are written back to disk only if they are changed in physical memory. A high rate of pages output might indicate a memory shortage.

Metrics

Windows NT writes more pages back to disk to free up space when low in physical memory. This counter counts numbers of pages, and can be compared to other counts of pages, without conversion. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

Although it never hurts to have as much physical memory as your system can handle, there are some things you can check within your system to alleviate the memory bottleneck.

- Check to see if you have any drivers or protocols that are running but not being used. They use space in all memory pools even if they are idle.
- Check to see if you have additional space on your disk drive that you could use to expand the size of your page file. Normally, the bigger the initial size of your page file, the better, in performance terms.

Free Physical

The Free Physical statistic is the amount of physical memory that is uncommitted.

Metrics

None.

Free Paged

The Free Paged statistic is the amount of uncommitted virtual memory.

Metrics

None.

Total Physical

The Total Physical statistic is the total physical memory available.

Metrics

None.

Total Paged

The Total Paged statistic is the amount of total virtual memory, in bytes. Used Memory is the physical memory that has space reserved on the disk paging file(s). There can be one or more paging files on each physical drive.

Metrics

None.

Page Faults/Sec

The Page Faults/Sec statistic is the overall rate faulted pages are handled by the processor. It is measured in numbers of pages faulted per second. A page fault occurs when a process requires code or data that is not in its working set. This counter includes both hard faults and soft faults.

Metrics

This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval.

Troubleshooting

If the number of page faults remains consistently high, you can check with your Windows System Administrator for further investigation. Often, large numbers of page faults are not a problem so long as they are soft faults. However, hard faults, that require disk access, can cause delays.

Cache Efficiency

The Cache Efficiency section of the Memory tab succinctly communicates the general overall performance levels of the server's memory. The following statistics are available in this section:

- Copy Read Hits%
- Data Map Hits%
- MDL Read Hits%
- Pin Read Hits%

Copy Read Hits %

The Copy Read Hits % statistic is the percentage of cache copy read requests that hit the cache and does not require a disk read to provide access to the page in the cache.

Metrics

When the page is pinned in the memory, the page's physical address in the file system cache will not be altered. A copy read is a file read operation where a page in the cache is copied to the application's buffer. Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate

Data Map Hits %

The Data Map Hits % statistic is the percentage of data maps in the file system cache that could be resolved without having to retrieve a page from the disk.

Metrics

Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate.

MDL Read Hits %

The MDL Read Hits % statistic is the percentage of Memory Descriptor List Read requests to the file system cache that hit the cache and does not require disk access to provide memory access to the pages in the cache.

Metrics

Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate.

Pin Read Hits %

The Pin Read Hits % statistic is the percentage of pin read requests that hit the file system cache and does not require a disk read in order to provide access to the page in the file system cache.

Metrics

Because this value reflects hits, it ideally should be close to 100%. An amount below 100% indicates misses.

Troubleshooting

Adding physical memory to a server results in a larger file system cache, which is generally more efficient. Defragmenting disks also helps, putting related pages in the cache together and thereby improving the cache hit rate.

Space Tab

The Space tab of the OS Detail page includes the following sections:

- Disk Space Free
- Disk Space Detail

Disk Space Free

The Disk Space Free metric displays the amount of free disk space in megabytes.

Metric

None.

Disk Space Detail

The Disk Space Detail section of the Space tab succinctly communicates the general overall performance levels of the server’s disks and space allotment. The table below describes the statistics in this section:

Statistic	Description
Partition	The drive letter of the disk.
Total Space	Total size of the disk/device's capacity expressed in MBs.
Used Space	Amount of MBs currently allocated on the particular disk/device.
Free Space	Amount of MBs currently unallocated and free on the particular disk/device.

Network Tab

The Network tab of the OS Detail page succinctly communicates the general overall performance levels of the server's networking. The statistics available in this section depend on the platform of operating system. The table below describes the information available in this section:

Column	Description
Network Interface	The name of network interface.
INET Address/Address	The IP address assigned to the network interface.
Pkts Sent/Sec	The number of packets sent per second.
Pkts Received/Sec	The number of packets received per second.
Sent (KB/Sec)	The number of bytes sent per second.
Received (KB/Sec)	The number of bytes received per second.
Out Pkts Discarded	The number of outbound packets discarded.
In Pkts Discarded	The number of inbound packets discarded.
Out Pkt Errors	The number of outbound packet errors.
In Pkt Errors	The number of inbound packet errors.
Queue Length	The queue length.
Collisions	The number of collisions.
Packets Discarded	The number of packets discarded.

Session Detail View

The Session Detail view is available for the following database platforms:

- Oracle Session Detail View
- SQL Server Session Detail View
- Sybase Session Detail View
- DB2 Session Detail View

Oracle Session Detail View

The following tabbed pages are available on the Session Detail view for Oracle:

- Session Contention
- Session I/O
- Session Memory
- Session Network
- Session Objects
- Session SQL
- Session Statistics

Session Memory Tab for Oracle

- Metrics

The Session Memory tab of the Session Detail view presents the statistics surrounding a session's memory usage. The table below describes the information available on the Session Memory tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the memory related statistic.
Value	The cumulative value for the memory statistic.
Cache Hit Ratio	The percentage of data obtained from memory access vs. physical I/O.

Metrics

Sessions with abnormally high memory usage can affect overall performance at the server level, as this memory (PGA and UGA) is allocated outside of the Oracle SGA (unless the multi-threaded server option is being used). If cache hit ratios at the session level are lower than 85 percent, data access can be inefficient.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Session I/O Tab for Oracle

- Metrics

The Session I/O tab of the Session Detail view presents the statistics surrounding a session's I/O activity. The table below describes the information available on the Session I/O tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the I/O related statistic.
Value	The cumulative value for the I/O statistic.

Metrics

Seeing high values for physical reads and writes can indicate an inefficient session. Large numbers of physical reads can imply a session with too many large table scans or inefficient SQL operations. Large numbers of physical writes can be okay for sessions inputting large volumes of data into the database. Or, they could also indicate a session involved in heavy disk sort activity.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Session Contention Tab for Oracle

- Metrics

The Session Contention tab of the Session Detail view presents information relating to resources on which the current session is waiting. The first grid displays user waits. The second grid displays user locks. The table below describes the information available in the User Waits grid on the Session Contention tab of the Session Detail view for Oracle:

Column	Description
Wait Cause	The wait event being experienced by the session.
Program	The program the session is executing against Oracle.
Seconds	The number of seconds the session has spent in the wait.

Column	Description
State	The status of the wait event (WAITING, etc.)

The table below describes the information available in the User Locks grid on the Session I/O tab of the Session Detail view for Oracle:

Column	Description
User	The user account being used by the session.
Terminal	The client machine name used by the session.
SID	The unique Oracle identifier for the session.
Serial #	The serial number for the session.
Table	The object locked by the session.
Lock Mode	The lock mode used by the session.
Title	The lock request issued by the session.

Metrics

You can ignore some waits, like the SQL*Net more data from client and SQL*Net message from client. Others, like enqueue waits, can be indicative of a lock contention problem. Waits, like db file scattered reads, can indicate sessions involved in long table scan operations.

Locks that are held for unusually long periods require further investigation. The application logic can be inefficient or the program is not issuing COMMITs frequently enough. The culprit of blocking-lock scenarios is usually the application design, or the SQL used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the correct SQL to get the job done is an art. Most DBAs who have had to face Oracle Forms applications have suffered through the dreaded SELECT…FOR UPDATE statements. These place unnecessary restrictive locks on nearly every read operation. The key to avoiding lock contention is to process user transactions as quickly and efficiently as possible.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session Objects Tab for Oracle

- Metrics

The Session Objects tab of the Session Detail view presents information regarding the objects in use by a session. The table below describes the information available in the Objects Accessed grid on the Session Objects tab of the Session Detail view for Oracle:

Column	Description
Owner	The owner of the object.
Type	The type of object (TABLE, VIEW, etc.).
Object	The name of the object.

The Rollback Segments Accessed grid displays the names of rollback segments currently being used by the session.

Metrics

Once you have an idea of which objects your users access most often, you can refine some processes to facilitate access to them. You can use the Oracle 8 concept of the KEEP buffer cache to force Oracle to hold often-referenced data for data objects. The KEEP buffer cache is ideal for holding small look-up tables. If you have an earlier version of Oracle or do not want to split up your current buffer cache, you can use the CACHE table attribute to encourage Oracle to keep data blocks of CACHE'd tables at the most recently used end of the LRU buffer cache chain.

If you consistently see a session with active rollbacks, it can indicate locks are being held for long durations. It can also indicate that a session is using code without frequent enough COMMIT points.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session Network Tab for Oracle

- Metrics

The Session Network tab of the Session Detail view presents information about requests being sent to and from the database with respect to the current session. The table below describes the information available on the Session Network tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the SQL*Net related statistic.
Value	The cumulative value of the SQL*Net related statistic.

Metrics

None.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session SQL Tab for Oracle

- Metrics

The Session SQL tab of the Session Detail view presents information relating to any current SQL issued by the session as well as any SQL previously issued by the session.

Metrics

To determine access paths, you should export and run through an EXPLAIN PLAN session any SQL that you suspect of inefficient access.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

Session Statistics Tab for Oracle

- Metrics

The Session Statistics tab of the Session Detail view presents information relating to all recorded performance and miscellaneous statistics for the current session. The table below describes the information available on the Session Statistics tab of the Session Detail view for Oracle:

Column	Description
Statistic	The name of the session related statistic.

Column	Description
Value	The cumulative value of the session related statistic.

Metrics

None.

> **TIP:** To configure the grid to show/hide row numbers, use the Options Editor.

SQL Server Session Detail View

The Session Detail view down provides a granular look at the details when a process is acting in a way that merits further investigation. It also presents data about a particular process in a way that makes it easier to understand and view apart from all other processes that SQL Server is running.

The following tabbed pages are available on the Session Detail view for SQL Server:

- All Locks
- Blocked By
- Blocking
- Overview
- SQL

Overview Tab for SQL Server

- Metrics

The Overview tab of the Session Detail view displays information to analyze the details of a particular process. The tables below describe the statistics for each category on the Overview tab of the Session Detail view for SQL Server. The available categories are:

- Contention
- General
- I/O
- Memory
- Network
- Users

General Statistics

The table below describes the statistics in the General category on the Overview tab of the Session Detail view:

Statistic	Description
SPID	The SQL Server process ID. Unique value across all processes.
Login Name	The SQL Server login name of the process.
NT User	If using Windows Authentication, the name of the Windows NT user for this process. If using a trusted connection, the Windows NT user name.

Statistic	Description
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Database	The database in which the process is running.
Program	The executable the process is using against the server.
Host	The machine name that originated the process.

Memory Statistics

The table below describes the statistics in the Memory category on the Overview tab of the Session Detail view:

Statistic	Description
Memory Usage	The number of pages in the procedure cache (SQL Cache) that are currently allocated to this process. A negative number indicates that pages are being released (freed) from this process.
Buffer Cache Hit Ratio	The percentage of data page requests by this process that is available in memory as opposed to performing a physical I/O to disk.

Contention Statistics

The table below describes the statistics in the Contention category on the Overview tab of the Session Detail view:

Statistic	Description
Blocked By	If the process is being blocked, the SPID of the blocking process. A value of zero means that the process is not blocked.
Wait Time	The number of milliseconds that the process has been waiting to be serviced. A value of zero indicates that the process is not waiting.
Last Wait Type	The last or current SQL Server wait type.
Wait Resource	The SQL Server's textual representation of a lock resource.

I/O Statistics

The table below describes the statistics in the I/O category on the Overview tab of the Session Detail view:

Statistic	Description
Physical I/O	The number of physical and logical reads performed by this session.
Physical Reads	The number of physical reads from disk performed by this session.
Logical Reads	The number of reads from memory performed by this process.
Logical Writes	The number of writes to memory performed by this process.

Users Statistics

The table below describes the statistics in the Users category on the Overview tab of the Session Detail view:

Statistic	Description
CPU Usage	The cumulative CPU time for the process. The CPU usage is updated regardless of the value of the SET STATISTICS TIME ON option.
Open Trans	The current number of open transactions owned by the process.
Login Time	For client process, the last time a client logged into the SQL Server instance. For system processes, the time that SQL Server was started.

Statistic	Description
Last Batch	For client process, the last time a remote stored procedure call or an EXECUTE statement was executed. For system processes, the time that SQL Server was started.

Network Statistics

The table below describes the statistics in the Network category on the Overview tab of the Session Detail view:

Statistic	Description
Net Address	The unique identifier of the network card in the client machine that owns the process.
Net Library	When a process is initiated from a client, the controlling mechanism is the network connection. Each network connection has a library associated with it. This value is the name library associated with the network connection responsible for this process.

Metrics

High memory usage and a low cache hit ratio for a given process over a sustained period of time could indicate that the process is using poorly written code. Check the SQL tab to investigate further.

Also, watch for an unusually high percentage of CPU use over a long period of time. This could indicate a rogue process that must be terminated by the DBA using a KILL command for the session.

SQL Tab for SQL Server

- Metrics

The SQL tab of the Session Detail view presents information relating to any current SQL issued by the session as well as any SQL previously issued by the session.

Metrics

To determine access paths, you should export and run through a QUERY PLAN session, any SQL suspect of inefficient access.

Blocked By Tab for SQL Server

- Metrics
- Troubleshooting

Without a doubt, blocking lock situations can give the appearance of a “frozen” database almost more than anything else. A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches on large systems. Although SQL Server supports row-level locking, blocking lock situations do crop up - sometimes frequently.

Metrics

The Blocked By tab contains information relating to processes that are currently blocking the process featured in the Session Detail view. The table below describes the information available on the Blocked By tab of the Session Detail view for SQL Server:

Column	Description
SPID	The SQL Server process ID. It is a unique value across all processes.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Login	The SQL Server login name of the process.
NT User	If using Windows Authentication, the name of the Windows NT user for this process. If using a trusted connection, the Windows NT user name.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Memory	The number of pages in the procedure cache (SQL Cache) that are currently allocated to this process. A negative number indicates that pages are being released (freed) from this process.
CPU	The cumulative CPU time for the process. The CPU usage is updated regardless of the value of the SET STATISTICS TIME ON option.
Physical I/O	The current cumulative number of physical disk reads and writes issued by the process.
Blocked	If the process is being blocked, the SPID of the blocking process.
Database	The database in which the process is running.
Command	The current command being executed by the process.
Last Batch	For a client process, the last time a remote stored procedure call or an EXECUTE statement was executed. For system processes, the time that SQL Server was started.
Login Time	For a client process, the last time a client logged into the SQL Server instance. For system processes, the time that SQL Server was started.
Wait Time	The time that the process has been waiting to be serviced, in milliseconds. A value of zero indicates that the process is not waiting.
Wait Type	The last or current SQL Server wait type.
Open Xacts	The current number of open transactions owned by the process.
NT Domain	If using Windows Authentication or a trusted connection, the name of the Windows domain of the user who owns the process.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied - the DBA issues a KILL against the offending process, which eliminates the user's stranglehold on the objects the user was accessing. Other user processes then nearly almost always complete in an instant. Discovering the blocked lock situation is made easier by using tools like Performance Center, but preventing the blocking lock situation in the first place is tricky.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

By default, all processes wait indefinitely for locks in SQL Server. You can change this behavior by using the SET LOCK_TIMEOUT command, which limits the number of seconds that a process waits for a lock before timing out.

Blocking Tab for SQL Server

Without a doubt, blocking lock situations can give the appearance of a “frozen” database almost more than anything else. A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches on large systems. Although SQL Server supports row-level locking, blocking lock situations do crop up - sometimes frequently.

Metrics

The Blocking tab contains information on blocks issued by the featured process that are blocking other processes. The table below describes the information available on the Blocking tab of the Session Detail view for SQL Server:

Column	Description
SPID	The SQL Server process ID. It is a unique value across all processes.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Login	The SQL Server login name of the process.
NT User	If using Windows Authentication, the name of the Windows NT user for this process. If using a trusted connection, the Windows NT user name.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Memory	The number of pages in the procedure cache (SQL Cache) that are currently allocated to this process. A negative number indicates that pages are being released (freed) from this process.
CPU	The cumulative CPU time for the process. The CPU usage is updated regardless of the value of the SET STATISTICS TIME ON option.
Physical I/O	The current cumulative number of physical disk reads and writes issued by the process.
Database	The database in which the process is running.
Command	The current command being executed by the process.
Last Batch	For client process, this value represents the last time a remote stored procedure call or an EXECUTE statement was executed. For system processes, it represents the time at which SQL Server was started.
Login Time	For client process, the last time a client logged into the SQL Server instance. For system processes, the time that SQL Server was started.
Wait Time	The time that the process has been waiting to be serviced, in milliseconds. A value of zero indicates that the process is not waiting.
Wait Type	The last or current SQL Server wait type.
Open Xacts	The current number of open transactions owned by the process.
NT Domain	If using Windows Authentication or a trusted connection, the name of the Windows Domain of the user who owns the process.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied - the DBA issues a KILL against the offending process, which eliminates the user's stranglehold on the objects the user was accessing. Other user processes then nearly almost always complete in an instant. Discovering the blocked lock situation is made easier by using tools like Performance Center, but preventing the blocking lock situation in the first place is tricky.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

By default, all processes wait indefinitely for locks in SQL Server. You can change this behavior by using the SET LOCK_TIMEOUT command, which limits the number of seconds that a process waits for a lock before timing out.

All Locks Tab for SQL Server

- Metrics
- Troubleshooting

Without a doubt, blocking lock situations can give the appearance of a “frozen” database almost more than anything else. A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches even on large systems. Although SQL Server supports row-level locking, blocking lock situations do crop up - sometimes frequently.

Metrics

The All Locks tab contains information on all locks associated with the featured SPID, including locks that are held by the process and locks that are blocking the process. The table below describes the information available on the All Locks tab of the Session Detail view for SQL Server:

Column	Description
Database	The database where the locks are occurring.
Table Name	The name of the table involved in a lock. NULL for non-table locks or table locks that take place in the tempdb database.
Ndx ID	The index ID involved in the lock.
Lock Type	The type of lock (database, table, row ID, etc.).
Lock Mode	The mode of the lock (shared, exclusive, etc.).
Lock Status	The status of the lock (waiting or granted).
Owner Type	Whether the lock came from a regular session or a transaction.
Program	The executable the process is using against the server.
BLK SPID	If nonzero, the process ID of the process blocking the requested lock. A value of zero indicates that the process is not blocked.
Wait Time	The time the process has waited for the lock, in milliseconds.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Command	The command the process is currently issuing.
NT Domain	The name of the Windows 2000/NT domain.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied - the DBA issues a KILL against the offending process, which eliminates the user's stranglehold on the objects the use was accessing. Other user processes then nearly almost always complete in an instant. Discovering the blocked lock situation is made easier by using tools like Performance Center, but preventing the blocking lock situation in the first place is tricky.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

By default, all processes wait indefinitely for locks in SQL Server. You can change this behavior by using the SET LOCK_TIMEOUT command, which limits the number of seconds that a process waits for a lock before timing out.

Sybase Session Detail View

The Session Detail view down provides a granular look at the details when a process is acting in a way that merits further investigation. It also presents data about a particular process in a way that makes it easier to understand and view apart from all other processes that Sybase is running.

The following tabbed pages are available on the Session Detail view for Sybase:

- Overview
- SQL
- Blocked By
- Blocking
- All Locks

Overview Tab for Sybase

The Overview tab of the Session Detail view displays information to analyze the details of a particular process. The tables below describe the statistics for each category on the Overview tab of the Session Detail view for Sybase. The available categories are:

- Contention
- Execution
- General
- I/O
- Memory
- Users

General Statistics

The table below describes the statistics in the General category on the Overview tab of the Session Detail view:

Statistic	Description
SPID	The Sybase process ID. Unique value across all processes.
Login Name	The Sybase login name of the process.
Family ID	The ID of the coordinating process and all of its worker processes.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Database	The database in which the process is running.
Program	The executable the process is using against the server.

Statistic	Description
Host	The machine name that originated the process.

Memory Statistics

The Memory Usage statistic in the Memory category on the Overview tab of the Session Detail view is the number of pages in the procedure cache (SQL Cache) that are currently allocated to this process. A negative number indicates that pages are being released (freed) from this process.

Contention Statistics

The table below describes the statistics in the Contention category on the Overview tab of the Session Detail view:

Statistic	Description
Blocked By	If the process is being blocked, the SPID of the blocking process. A value of zero indicates that the process is not blocked.
Time Blocked	The SQL Server's wait time.

I/O Statistics

The Physical I/O statistic in the I/O category on the Overview tab of the Session Detail view is the number of physical and logical reads performed by this session.

Users Statistics

The table below describes the statistics in the Users category on the Overview tab of the Session Detail view:

Statistic	Description
CPU Usage	The cumulative CPU time for the process. The CPU usage is updated regardless of the value of the SET STATISTICS TIME ON option.
Active Trans	The current number of open transactions owned by the process.

Execution Statistics

The table below describes the statistics in the Execution category on the Overview tab of the Session Detail view:

Statistic	Description
Exec Class	The execution class for the current process.
Priority	The priority of the current process.
Affinity	The affinity level for the current process.

SQL Tab for Sybase

The SQL tab of the Session Detail view presents information relating to any current SQL issued by the session as well as any SQL previously issued by the session.

Metrics

To determine access paths, you should export and run through a QUERY PLAN session, any SQL suspect of inefficient access.

Blocked By Tab for Sybase

- Metrics
- Troubleshooting

Without a doubt, blocking lock situations can give the appearance of a “frozen” database almost more than anything else. A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches even on large systems.

Metrics

The Blocked By tab contains information relating to processes that are currently blocking the process featured in the Session Detail view. The table below describes the information available on the Blocked By tab of the Session Detail view for Sybase:

Column	Description
SPID	The Sybase process ID. This is a unique value across all processes.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Login	The Sybase login name of the process.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Memory	The number of pages in the procedure cache (SQL Cache) that are currently allocated to this process. A negative number indicates that pages are being released (freed) from this process.
CPU	The cumulative CPU time for the process. The CPU usage is updated regardless of the value of the SET STATISTICS TIME ON option.
Physical I/O	The current cumulative number of physical disk reads and writes issued by the process.
Database	The database in which the process is running.
Command	The current command being executed by the process.
Time Blocked	The time that the process has been blocked, in seconds.
Transaction	The name of any transaction.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied - the DBA issues a KILL against the offending process, which eliminates the user's stranglehold on the objects the user was accessing. Other user processes then nearly almost always complete in an instant. Discovering the blocked lock situation is made easier by using tools like Performance Center, but preventing the blocking lock situation in the first place is tricky.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

Blocking Tab for Sybase

- Metrics
- Troubleshooting

Without a doubt, blocking lock situations can give the appearance of a “frozen” database almost more than anything else. A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches on large systems.

Metrics

The Blocking tab contains information on blocks issued by the featured process that are blocking other processes. The table below describes the information available on the Blocking tab of the Session Detail view for Sybase:

Column	Description
SPID	The Sybase process ID. This is a unique value across all processes.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Login	The Sybase login name of the process.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Memory	The number of pages in the procedure cache (SQL Cache) that are currently allocated to this process. A negative number indicates that pages are being released (freed) from this process.
CPU	The cumulative CPU time for the process. The CPU usage is updated regardless of the value of the SET STATISTICS TIME ON option.
Physical I/O	The current cumulative number of physical disk reads and writes issued by the process.
Database	The database in which the process is running.
Command	The current command being executed by the process.
Time Blocked	The time that the process has been blocked, in seconds.
Transaction	The name of any transaction.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied - the DBA issues a KILL against the offending process, which eliminates the user's stranglehold on the objects they were accessing. Other user processes then nearly almost always complete in an instant. Discovering the blocked lock situation is made easier by using tools like Performance Center, but preventing the blocking lock situation in the first place is tricky.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

All Locks Tab for Sybase

Without a doubt, blocking lock situations can give the appearance of a “frozen” database almost more than anything else. A single blocking user has the potential to stop work for nearly all other processes on a small system, and can cause major headaches on large systems.

Metrics The All Locks tab contains information on all locks associated with the featured SPID, including locks that are held by the process and locks that are blocking the process. The table below describes the information available on the All Locks tab of the Session Detail view for Sybase:

Column	Description
Database	The database where the locks are occurring.
Lock Type	The type of lock (database, table, row ID, etc.)
Object Name	The name of the object being blocked.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Lock Page	The page number where the lock is currently applied.
Lock Class	The name of the cursor the lock is associated with (if any).
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Command	The command the process is currently issuing.
Transaction	The name of any transaction.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Troubleshooting

Once discovered, a blocking lock situation can normally be quickly remedied—the DBA issues a KILL against the offending process, which eliminates the user's stranglehold on the objects the user was accessing. Other user processes then nearly almost always complete in an instant. Discovering the blocked lock situation is made easier by using tools like Performance Center, but preventing the blocking lock situation in the first place is tricky.

The culprit of blocking lock scenarios is usually the application design, or the SQL being used within the application itself. Properly coding an application to reference database objects in an efficient order, and then using the right SQL to get the job done, is an art. The key to avoiding lock contention is to process user transactions in the quickest and most efficient manner possible - something not always easy to do.

DB2 Session Detail View

The following tabbed pages are available on the Session Detail view for DB2:

- Application Tab for DB2
- Unit of Work Tab for DB2
- Locking Tab for DB2
- Memory Tab for DB2
- I/O Tab for DB2
- SQL Statistics Tab for DB2

The following information about the selected application is available on each page:

Field	Description
Agent ID	This is a system-wide unique indentifier for the application.
Auth ID	This is the authorization ID of the user who invoked the application being monitored.

Field	Description
OS User ID	This is the authorization ID used to access the operating system.
Application	This is the name of the application executable.

Application Tab for DB2

The Application tab contains application and client detail information from the application snapshot record for the selected application.

Application Details

Statistic	Description
Application Agent ID	This is a system-wide unique indentifier for the application.
Application Agent PID	The process ID (UNIX) or thread ID (Windows) of a DB2 agent.
Application Agent System CPU (sec)	This is the total system CPU time (in seconds) used by the database manager agent process.
Application Agent User CPU (sec)	This is the total CPU time (in seconds) used by database manager agent process.
Application Agents	This is the number of agents currently executing statements or subsections.
Application Agents Stolen	This is the number of times agents are stolen from the application. Agents are stolen when an idle agent is reassigned from one application to another.
Application Assoc Agents HWM	This is the maximum number of subagents associated with the application.
Application Assoc Sub-Agents	This is the number of subagents associated with the application.
Application Authorization ID	This is the authorization ID of the user who invoked the application being monitored.
Application Codepage	This is the code page identifier.
Application Connect Complete Time	This is the date and time a connection request was granted.
Application Connect Start Time	This is the date and time an application started a connection request.
Application Coord Agent PID	This is the process ID (UNIX) or thread ID (Windows) of the coordinator agent for the selected application.
Application Coord Node	In a multi-node system, this is the node number of the node where the selected application connected or attached to the instance.
Application Coord Token	This is the DRDA AS correlation token.
Application ID	This is generated when the application connects to the database at the database manager or when DDCS receives a request to connect to a DRDA database.
Application Idle Time	This is the number of seconds since the selected application last issued any requests to the server. This includes applications that have not terminated a transaction, for example not issued a commit or rollback.
Application Name	Name of the application executable.
OS User ID	The authorization ID used to access the operating system.
Application Priority	This is the priority of the agents working for this application.
Application Priority Type	This is the operating system priority type for the agent working on behalf of the application.
Application Sequence Number	This identifier is incremented when a unit of work ends (when a COMMIT or ROLLBACK terminates a unit of work). Together, the application ID and application sequence number uniquely identify a transaction.

Statistic	Description
Application Session Auth ID	This is the current authorization ID for the session being used by this application.
Application Status	This is the current status of the application.
Application Status Change Time	This is the date and time the application entered its current status.
Application Territory Code	This is the territory code (formerly country code) of the database for which the monitor data is being collected.

Application Authorities

Column	Description
Authority	This is the highest authority level granted to the application.
Explicit	Authorizations granted explicitly to a user.
Indirect	Indirect authorizations inherited from group or public.

Client Details

Statistic	Description
Client Database Alias	This is the alias of the database provided by the application to connect to the database.
Client Inbound Comm Address	This is the communication address of the client.
Client Node Name	This is the node name (nname) in the database manager configuration file at the client database partition.
Client PID	This is the process ID of the client application that made the connection to the database.
Client Platform	This is the operating system on which the client application is running.
Client Product and Version	This is the product and version that is running on the client.
Client Protocol	This is the communication protocol that the client application is using to communicate with the server.
TP Client Accounting String	This is the data passed to the target database for logging and diagnostic purposes (if the sqleseti API was issued in this connection).
TP Client Application Name	This name identifies the server transaction program performing the transaction (if the sqleseti API was issued in this connection).
TP Client User ID	This is the client user ID generated by a transaction manager and provided to the server (if the sqleseti API is used).
TP Client Workstation Name	This name identifies the client's system or workstation, for example CICS EITERMID, if the sqleseti API was issued in this connection.

Unit of Work Tab for DB2

The Unit of Work tab contains the SQL statement for the selected application, as well as statistics related to that statement.

SQL Statement

The SQL Statement field displays the SQL statement for the application. You can click **Explain SQL** to view an explain plan for the statement.

Statement Detail

Statistic	Description
SQL Statement	This is the text of the dynamic SQL statement.
Node	This is the node where the statement was executed.
Type	This is the type of statement processed.
Creator	This is the authorization ID of the user that pre-compiled the application.
Operation	This is the statement operation currently being processed or that was most recently processed (if none are currently running).
Agents Created	This is the maximum number of agents that were used when executing the statement.
Agents Working	This is the number of concurrent agents currently executing a statement or subsection.
Cursor Name	This is the name of the cursor corresponding to this SQL statement.
Blocking Cursor	This indicates if the statement being executed is using a blocking cursor.
Package Name	This is the name of the package that contains the SQL statement that is currently executing.
Package Version	This identifies the version identifier of the package that contains the currently executing SQL statement.
Section Number	This is the internal section number in the package for the SQL statement currently processing or most recently processed.
Parallelism Degree	This is the degree of parallelism requested when the query was bound.
Query Cost Estimate	This is the estimated cost for a query (in timerons) as determined by the SQL compiler.
Query Card Estimate	This is an estimate of the number of rows that will be returned by a query.

Statement Statistics

Statistic	Description
Statement Start Time	This is the date and time when the statement operation (stmt_operation) monitor element started executing.
Statement Stop Time	This is the date and time when the statement operation (stmt_operation) monitor element stopped executing.
Statement Sort Time (sec)	This is the total elapsed time (in seconds) for all sorts that have been executed.
Statement User CPU (sec)	This is the total user CPU time (in seconds) used by the currently executing statement.
Statement System CPU (sec)	This is the total system CPU time (in seconds) used by the currently executing statement.
Statement Elapsed Time (sec)	This is the elapsed execution time (in seconds) of the most recently completed statement.
UOW Start Time	This is the date and time that the unit of work first required database resources.
UOW Stop Time	This is the date and time that the most recent unit of work completed. This occurs when database changes are committed or rolled back.
UOW Prev Stop Time	This is the time the previous unit of work completed.
UOW Elapsed Time (sec)	This is the elapsed execution time (in seconds) of the most recently completed unit of work.
UOW Lock Wait Time (sec)	This is the total amount of elapsed time (in seconds) this unit of work has spent waiting for locks.
UOW Log Space Used (KB)	This is the amount of log space (in kilobytes) used in the current unit of work of the monitored application.
UOW Completion Status	This is the status of the unit of work and how it stopped.

Statement Activity

Statistic	Description
Sorts	This is the total number of times a set of data was sorted in order to process the statement operation (stmt_operation).
Sort Overflows	This is the total number of sorts that ran out of sort heap and may have used disk space for temporary storage.
Fetches	This is the number of successful fetches that were performed on a specific cursor.
Rows Read	This is the number of rows read from the table.
Rows Written	This is the number of rows changed (inserted, deleted, or updated) in the table.
Internal Rows Deleted	This is the number of rows deleted from the database as a result of internal activity.
Internal Rows Updated	This is the number of rows updated from the database as a result of internal activity.
Internal Rows Inserted	This is the number of rows inserted into the database as a result of internal activity that was caused by triggers.
Temporary Data Logical Reads	This indicates the number of data pages that have been requested from the buffer pool (logical) for temporary table spaces. **NOTE:** The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.
Temporary Data Physical Reads	Indicates the number of data pages that have been read in from the table space containers (physical) for temporary table spaces. **NOTE:** The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.
Temporary Index Logical Reads	Indicates the number of index pages that have been requested from the buffer pool (logical) for temporary table spaces. **NOTE:** The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.
Temporary Index Physical Reads	Indicates the number of index pages that have been read in from the table space containers (physical) for temporary table spaces. **NOTE:** The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Locking Tab for DB2

The Locking tab is a single application view of the Lock View. Information that is found in the Locks Held and Locks Waiting tabs of the Lock View will be available here for the selected application.

Locks Held

Column	Description
Lock Mode	This is the type of lock being held.
Object Type	This is the type of object against which the application holds a lock (for object-lock-level information), or the type of object for which the application is waiting to obtain a lock (for application-level and deadlock-level information).
Table Schema	This is the schema of the table that the lock is on.
Table Name	This is the name of the table that the lock is on. This element is only set if Object Type indicates Table.
Tablespace	This is the name of the table space against which the lock is held.
Lock Status	This is the lock's status (waiting or granted).

Column	Description
Escalation	This indicates whether a lock request was made as part of a lock escalation.

Locks Waiting

Column	Description
Agent ID	This is the application handle of the agent holding a lock for which this application is waiting.
Application ID	This is the application ID of the application that is holding a lock on the object that this application is waiting to obtain.
Lock Mode	This is the type of lock being held.
Mode Requested	This is the lock mode being requested by the application.
Object Type	This is the type of object against which the application holds a lock.
Table Schema	This is the schema of the table that the lock is on.
Tablespace	This is the name of the table that the lock is on. This element is only set if Object Type indicates Table.
Wait Start Time	This is the date and time that this application started waiting to obtain a lock on the object that is currently locked by another application.
Escalation	This indicates whether a lock request was made as part of a lock escalation.

Memory Tab for DB2

The Memory tab displays all the memory pools for the application and key cache statistics specific to the application.

Memory Pools

Column	Description
Node	This is the number assigned to the node in the db2nodes.cfg file.
Memory Pool	This is the type of memory pool.
Utilization	This is the percentage of memory pool used.
High Watermark (MB)	This is the largest size of a memory pool (in megabytes) since its creation.
Current Size (MB)	This is the current size of a memory pool (in megabytes).
Max Size (MB)	This is the internally configured size of a memory pool (in megabytes) in DB2.

Memory Statistics

Statistic	Description
Application Section Inserts	This is the number of inserts of SQL sections by an application from its SQL work area.
Application Section Lookups	This is the number of lookups of SQL sections by an application from its SQL work area.
Catalog Cache Inserts	This is the number of times that the system tried to insert table descriptor or authorization information into the catalog cache.
Catalog Cache Lookups	This is the number of times that the catalog cache was referenced to obtain table descriptor information or authorization information.
Catalog Cache Overflows	This is the number of times that the catalog cache overflowed the bounds of its allocated memory.

Statistic	Description
Package Cache Inserts	This is the total number of times that a requested section was not available for use and had to be loaded into the package cache. This includes any implicit prepares performed by the system.
Package Cache Lookups	This is the number of times an application looked for a section or package in the package cache.
Private Workspace Inserts	This is the number of inserts of SQL sections by an application into the private workspace.
Private Workspace Lookups	This is the number of lookups of SQL sections by an application in its agents' private workspace.
Private Workspace Overflows	This is the number of times that the private workspaces overflowed the bounds of its allocated memory.
Private Workspace HWM (KB)	This is the largest size (in kilobytes) reached by the private workspace.
Shared Workspace Inserts	This is the number of inserts of SQL sections by applications into shared workspaces.
Shared Workspace Lookups	This is the number of lookups of SQL sections by applications in shared workspaces.
Shared Workspace Overflows	This is the number of times that shared workspaces overflowed the bounds of their allocated memory.
Shared Workspace HWM (KB)	This is the largest size (in kilobytes) reached by shared workspaces.

Cache Hit Ratios

Ratio	Description
Catalog Cache	The catalog cache hit ratio indicates how well the catalog cache is helping to avoid actual accesses to the catalog on disk. A high ratio indicates successful avoidance of actual disk I/O accesses.
Package Cache	The package cache hit ratio indicates how well the package cache is helping to avoid reloading packages and sections for static SQL from the system catalogs as well as helping to avoid recompiling dynamic SQL statements. A high ratio indicates successful avoidance of these activities.
Shared Workspace	The shared workspace hit ratio indicates how well the shared SQL workspace is helping to avoid having to initialize sections for SQL statements that are about to be executed. A high ratio indicate successful avoidance of this action.
Private Workspace	The private workspace hit ratio indicates how well the private SQL workspace is helping to avoid having to initialize sections for SQL statements that are about to be executed. A high ratio indicate successful avoidance of this action.
Application Workspace	The application workspace hit ratio indicates how well the application SQL workspace is helping to avoid having to initialize sections for SQL statements that are about to be executed. A high ratio indicate successful avoidance of this action.

I/O Tab for DB2

The I/O tab displays detailed I/O information from the application snapshot record for the selected application.

I/O Statistics

Statistic	Description
Buffered Data Logical Reads	This indicates the number of data pages that have been requested from the buffer pool (logical) for regular and large table spaces.

Statistic	Description
Buffered Data Physical Reads	This indicates the number of data pages that have been read from the table space containers (physical) for regular and large table spaces.
Buffered Data Writes	This indicates the number of times a buffer pool data page was physically written to disk.
Buffered Index Logical Reads	This indicates the number of index pages that have been requested from the buffer pool (logical) for regular and large table spaces.
Buffered Index Physical Reads	This indicates the number of index pages that have been read from the table space containers (physical) for regular and large table spaces.
Buffered Index Writes	This indicates the number of times a buffer pool index page was physically written to disk.
Buffered Read Time (sec)	This indicates the total amount of time (in seconds) spent reading data and index pages from the table space containers (physical) for all types of table spaces.
Buffered Write Time (sec)	This indicates the total amount of time (in seconds) spent physically writing data or index pages from the buffer pool to disk.
Direct Reads	This is the number of read operations that do not use the buffer pool.
Direct Read Requests	This is the number of requests to perform a direct read of one or more sectors of data.
Direct Writes	This is the number of write operations that do not use the buffer pool.
Direct Write Requests	This is the number of requests to perform a direct write of one or more sectors of data.
Direct Read Time (sec)	This is the elapsed time (in seconds) required to perform the direct reads.
Direct Write Time (sec)	This is the elapsed time (in seconds) required to perform the direct writes.
Extended Storage – Data Pages From	This is the number of buffer pool data pages copied from extended storage.
Extended Storage – Data Pages To	This is the number of buffer pool data pages copied to extended storage.
Extended Storage – Index Pages From	This is the number of buffer pool index pages copied from extended storage.
Extended Storage – Index Pages To	This is the number of buffer pool index pages copied to extended storage.
Prefetch Pages Unread	This indicates the number of pages that the prefetcher read in that were never used.
Prefetch Wait Time (sec)	This is the time an application spent waiting for an I/O server (prefetcher) to finish loading pages into the buffer pool.
Temporary Data Logical Reads	This indicates the number of data pages which have been requested from the buffer pool (logical) for temporary table spaces.
Temporary Data Physical Reads	This indicates the number of data pages read in from the table space containers (physical) for temporary table spaces.
Temporary Index Logical Reads	This indicates the number of index pages which have been requested from the buffer pool (logical) for temporary table spaces.
Temporary Index Physical Reads	This indicates the number of index pages read in from the table space containers (physical) for temporary table spaces.

I/O Distribution Ratios

Ratio	Description
Direct Read Ratio	Direct Reads are read operations that do not use the buffer pool. Direct Read Ratio is the percentage of all reads that were direct reads.

Ratio	Description
Logical Read Ratio	Logical Reads is the sum of all Buffer Pool Data Logical Reads and Buffer Pool Index Logical Reads. Logical Read Ratio is the percentage of all reads that were logical reads.
Physical Read Ratio	Physical Reads is the sum of all Buffer Pool Data Physical Reads and Buffer Pool Index Physical Reads. Physical Read Ratio is the percentage of all reads that were physical reads.
Direct Write Ratio	Direct Writes are write operations that do not use the buffer pool. Direct Write Ratio is the percentage of all writes that were direct writes.
Buffered Write Ratio	Buffered Writes is the sum of all Buffer Pool Data Writes and Buffer Pool Index Writes. Buffered Write Ratio is the percentage of all writes that were buffered writes.

SQL Statistics Tab for DB2

The SQL Statistics tab contains statistics about the SQL statement for the selected application.

SQL Statistics

Statistic	Description
Binds and Pre-Compiles	This is the number of binds and pre-compiles attempted.
Cursor Block Requests Accepted	This is the number of times a request for an I/O block was accepted.
Cursor Block Requests Rejected	This is the number of times a request for an I/O block at the server was rejected and the request was converted to non-blocked I/O.
Cursors Open Local	This is the number of local cursors currently open for this application, including those cursors counted by Cursors Open Local with Blocking.
Cursors Open Local with Blocking	This is the number of local blocking cursors currently open for this application.
Cursors Open Remote	This is the number of remote cursors currently open for this application, including those cursors counted by Cursors Open Remote with Blocking.
Cursors Open Remote with Blocking	This is the number of remote blocking cursors currently open for this application.
Internal Automatic Rebinds	This is the number of automatic rebinds (or recompiles) that have been attempted.
Internal Commits	This is the total number of commits initiated internally by the database manager.
Internal Rollbacks	This is the total number of rollbacks initiated internally by the database manager.
Internal Deadlock Rollbacks	This is the total number of forced rollbacks initiated by the database manager due to a deadlock. A rollback is performed on the current unit of work in an application selected by the database manager to resolve the deadlock.
Internal Rows Deleted	This is the number of rows deleted from the database as a result of internal activity.
Internal Rows Inserted	This is the number of rows inserted into the database as a result of internal activity caused by triggers.
Internal Rows Updated	This is the number of rows updated from the database as a result of internal activity.
Rows Deleted	This is the number of row deletions attempted.
Rows Inserted	This is the number of row insertions attempted.
Rows Read	This is the number of rows read from the table.
Rows Selected	This is the number of rows that have been selected and returned to the application.
Rows Updated	This is the number of row updates attempted.
Rows Written	This is the number of rows changed (inserted, deleted, or updated) in the table.

Statistic	Description
SQL DDL Statements	This indicates the number of SQL Data Definition Language (DDL) statements that were executed.
SQL Commit Statements	This indicates the total number of SQL COMMIT statements that have been attempted.
SQL Dynamic Statements	This indicates the number of dynamic SQL statements that were attempted.
SQL Failed Statements	This indicates the number of SQL statements that were attempted and failed.
SQL Requests Since Last Commit	This indicates the number of SQL requests submitted since the last commit.
SQL Rollback Statements	This indicates the total number of SQL ROLLBACK statements that have been attempted.
SQL Select Statements	This indicates the number of SQL SELECT statements that were executed.
SQL Static Statements	This indicates the number of static SQL statements that were attempted.
SQL UID Statements	This indicates the number of SQL UPDATE, INSERT, and DELETE statements that were executed.

SQL Distribution Ratios

Column	Description
DDL Statements	This is the percentage of all executed statements that were SQL DDL Statements.
UID Statements	This is the percentage of all executed statements that were SQL UID Statements.
Failed Statements	This is the percentage of all executed statements that were SQL Failed Statements.
Select Statements	This is the percentage of all executed statements that were SQL Select Statements.

Top Sessions View

The following tabbed pages are available on the Top Sessions view:

- Memory Tab
- I/O Tab
- CPU Tab

Memory Tab

The information on the Memory tab of the Top Sessions view depends on the target DBMS:

- Memory Tab for Oracle
- Memory Tab for SQL Server
- Memory Tab for Sybase
- Memory Tab for DB2

Memory Tab for Oracle

- Metrics

It is frequently the case that one or two users cause the majority of run-time problems. The problem could originate with a runaway process, an untuned batch procedure, or other user-initiated operation. User connections can get out of hand with memory consumption, and extreme cases have caused difficulties at both the database and operating system levels (ORA-4030 errors).

The table below describes the information available on the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view for Oracle:

Column	Description
User Name	The logon name the session is using.
SID	The unique Oracle identifier for the session.
Serial #	The serial number assigned to the session.
Status	The status of the session, ACTIVE or INACTIVE.
Machine	The name of the client machine name that the session is using.
PGA Memory (KB)	The Program Global Area (PGA) is a private memory area devoted to housing the global variables and data structures for a single Oracle process, in KB.
UGA Memory (KB)	The User Global Area (UGA) contains session specific information regarding open cursors, state information for packages, database link information, and more. When using Oracle's Multi-threaded Server (MTS), the UGA can be moved up into the SGA, in KB.
Memory Sorts (KB)	The total number of memory sorts a session has performed.
Total Memory (KB)	The memory (PGA + UGA) that the session is consuming, in KB.

NOTE: This information is available on both the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

If your database server does not have an overabundance of memory, you should check periodically to see who your heavy memory users are and the total percentage of memory each takes up. If you see that one or two users use more than 5-15 percent of the total memory, you should investigate the sessions further to see what activities they are performing.

Memory Tab for SQL Server

- Metrics

It is not uncommon for one or two users to cause the majority of runtime problems that plague a database. The problem could be a runaway process, an untuned batch procedure, or other user-initiated operation. User connections can get out of hand with memory consumption, and extreme cases have caused headaches at both the database and operating system levels.

Performance Center displays information to find processes that are using the most memory on the server on the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view.

The table below describes the information available on these tabs:

Column	Description
PID	The process ID of the connected session.
User Name	The user name assigned to the process.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Host	The machine name that originated the process.
Program	The program the process has invoked against SQL Server.
Memory Usage	The total number of memory pages allocated to the process.
Pct Mem Used	The percentage of overall memory among all processes that can be attributed to the process.
Database	The database in which the process is currently running.
Command	The command the process is currently issuing.
Open Trans	The number of open transactions for the process.
Blocked	The PID of any process blocking the current process.
Wait Time	The current wait time for the process, in milliseconds.

NOTE: This information is available on both the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

If your database server does not have an overabundance of memory, you should periodically check to see who your heavy memory users are along with the total percentage of memory each takes up. If you see one or two users who have more than 5-15 percent of the total memory usage, you should investigate the sessions further to see what activities they are performing.

Memory Tab for Sybase

- Metrics

It is not uncommon for one or two users to cause the majority of runtime problems that plague a database. The problem could be a runaway process, an untuned batch procedure, or other user-initiated operation. Often, user connections can get out of hand with memory consumption, and extreme cases have caused headaches at both the database and operating system levels.

This tab displays information to find processes that are using the most memory on the server. The table below describes the information available on the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view for Sybase:

Column	Description
PID	The process ID.
User Name	The user name assigned to the process.
FID	The process ID of the worker process' parent.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Memory Usage	The memory currently used by the process.
Pct Mem Used	The percentage of memory currently used by the process.
Database	The database in which the process is running.
Command	The command the process is currently issuing.
Transaction	The name of any transaction.
Blocked	The PID of any process blocking the current process.
Time Blocked	The time that the process has been blocked, in seconds.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

If your database server does not have an overabundance of memory, you should periodically check to see who your heavy memory users are along with the total percentage of memory each takes up. If you see one or two users who have more than 5-15 percent of the total memory usage, you should investigate the sessions further to see what activities they are performing.

Memory Tab for DB2

- Metrics

It is not uncommon for one or two users to cause the majority of runtime problems that plague a database. The problem could be a runaway process, an untuned batch procedure, or other user-initiated operation. Often, user connections can get out of hand with memory consumption, and extreme cases have caused headaches at both the database and operating system levels.

This tab displays information to find processes that are using the most memory on the server. The table below describes the information DB2 on the Leading Sessions tab of the Memory Detail view and the Memory tab of the Top Sessions view for DB2:

Column	Description
Agent ID	The application handle of the agent holding a lock for which this application is waiting.
Auth ID	The authorization ID of the user who invoked the application that is being monitored.
OS User ID	The operating system ID of the process.
Client PID	The process ID of the client application that made the connection to the database.
Application	The name of the application executable.
Status	The current status of the application.
UOW Elapsed Time (sec)	The elapsed execution time of the most recently completed unit of work.
Memory Overflows	Total number of memory overflows.
Memory Used (KB)	Total memory pool usage for the application.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

If your database server does not have an overabundance of memory, you should periodically check to see who your heavy memory users are along with the total percentage of memory each takes up. If you see one or two users who have more than 5-15 percent of the total memory usage, you should investigate the sessions further to see what activities they are performing.

I/O Tab

The information on the I/O tab of the Top Sessions view depends on the target DBMS:

- I/O Tab for DB2
- I/O Tab for SQL Server
- I/O Tab for Sybase
- I/O Tab for DB2

I/O Tab for Oracle

- Metrics

When a system undergoes heavy I/O activity, all the user connections can contribute somewhat equally to the overall load. Frequently, however, one or two user connections are responsible for 75 percent or more of the I/O activity. It may be that a large batch load or other typical process is running, and that is perfectly okay for your system. Or, it may be a runaway process or rogue connection that you should track down and eliminate.

It is a good idea to periodically check the leading sessions in your system with respect to I/O and make sure all is well. You can use Performance Center to perform this function with the information available on this tab. The table below describes the information available on the Leading Sessions tab of the I/O Detail view and the I/O tab of the Top Sessions view for Oracle:

Column	Description
User Name	The logon name the session is using.
SID	The unique Oracle identifier for the session.
Serial #	The serial number assigned to the session.
Status	The status of the session, ACTIVE or INACTIVE.
Machine	The name of the client machine name that the session is using.
Reads	The number of physical reads.
Writes	The number of physical writes.
Total I/O	The total of all physical I/O operations for the session.

NOTE: This information is available on both the Leading Sessions tab of the I/O Detail view and the I/O tab of the Top Sessions view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Pinpointing sessions with abnormally high I/O activity relative to other sessions in the system can aid in ferreting out accounts that are dragging down overall system performance. You should examine the activity of each session to determine the system workload and to see if you can reduce the workload or tune the system for better performance.

I/O Tab for SQL Server

- Metrics

When a system undergoes heavy I/O activity, sometimes you find that all the user connections are contributing somewhat equally to the overall load. Frequently, however, one or two user connections are responsible for 75 percent or more of the I/O activity. It can be that a large batch load or other typical process is running that is perfectly okay for your system. Alternatively, it can be a runaway process or other rogue connection that you should track down and possibly eliminate.

Performance Center displays information to find processes that are using the most memory on the server on the Leading Sessions tab of the I/O Detail view and the I/O tab of the Top Sessions view.

The table below describes the information available on these tabs:

Column	Description
PID	The process ID.
User Name	The user name assigned to the process.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Physical I/O	The current cumulative number of reads and writes issued by the process.
Pct I/O Used	The percentage of overall I/O that can be attributed to the process.
Database	The database in which the process is running.
Command	The command the process is currently issuing.
Open Trans	The number of open transactions for the process.
Blocked	The PID of any process blocking the current process.
Wait Time	The current wait time for the process, in milliseconds.

NOTE: This information is available on both the Leading Sessions tab of the I/O Detail view and the I/O tab of the Top Sessions view.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Finding one more two users that are consuming more than 75 percent of the total I/O load can indicate a runaway or improper process. By drilling down into the I/O activity of all users, you can quickly see if this is the case.

I/O Tab for Sybase

- Metrics

When a system undergoes heavy I/O activity, sometimes you find that all the user connections are contributing somewhat equally to the overall load. More often than not, however, one or two user connections are responsible for 75 percent or more of the I/O activity. It can be that a large batch load or other typical process is running that is perfectly okay for your system. Or it can be a runaway process or other rogue connection that needs to be tracked down and possibly eliminated.

It is a good idea to periodically check who the leading sessions are in your system with respect to I/O and make sure all is well. You can use Performance Center to easily perform this function with the leading sessions tab of the I/O Detail view. The table below describes the information available on the I/O tab of the Top Sessions view for Sybase:

Column	Description
PID	The process ID.
User Name	The user name assigned to the process.
FID	The process ID of the worker process' parent.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
Physical I/O	The current cumulative number of reads and writes issued by the process.
Pct I/O Used	The percentage of overall I/O that can be attributed to the process.
Database	The database in which the process is running.
Command	The command the process is currently issuing.
Transaction	The name of any transaction.
Blocked	The PID of any process blocking the current process.
Time Blocked	The time that the process has been blocked, in seconds.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Pinpointing sessions with abnormally high I/O activity relative to other sessions in the system help you ferret out accounts dragging down overall system performance. You should examine the activity of each session to determine the workload being placed on the system and if you can reduce or tune that workload for better performance.

I/O Tab for DB2

- Metrics

When a system undergoes heavy I/O activity, sometimes you find that all the user connections are contributing somewhat equally to the overall load. More often than not, however, one or two user connections are responsible for 75 percent or more of the I/O activity. It can be that a large batch load or other typical process is running that is perfectly okay for your system. Or it can be a runaway process or other rogue connection that needs to be tracked down and possibly eliminated.

It is a good idea to periodically check who the leading sessions are in your system with respect to I/O and make sure all is well. You can use Performance Center to easily perform this function with the leading sessions tab of the I/O Detail view. The table below describes the information available on the I/O tab of the Top Sessions view for DB2:

Column	Description
Agent ID	The application handle of the agent holding a lock for which this application is waiting.
Auth ID	The authorization ID of the user who invoked the application that is being monitored.
OS User ID	The operating system ID of the process.
Client PID	The process ID of the client application that made the connection to the database.
Application	The name of the application executable.
Status	The current status of the application.
UOW Elapsed Time (sec)	The elapsed execution time of the most recently completed unit of work.
Buffered I/O Time (ms)	The total time spent by application in performing buffered reads and writes.
Direct I/O Time (ms)	The total time spent by application in performing non-buffered reads and writes.
Total I/O Time (ms)	The total time spent by application in performing buffered and non-buffered reads and writes.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

Metrics

Pinpointing sessions with abnormally high I/O activity relative to other sessions in the system help you ferret out accounts dragging down overall system performance. You should examine the activity of each session to determine the workload being placed on the system and if you can reduce or tune that workload for better performance.

CPU Tab

The information on the CPU tab of the Top Sessions view depends on the target DBMS:

- CPU Tab for Oracle
- CPU Tab for SQL Server
- CPU Tab for Sybase
- CPU Tab for DB2

CPU Tab for Oracle

The table below describes the information available on the CPU tab of the Top Sessions view for Oracle:

Column	Description
User Name	The logon name the session is using.
SID	The unique Oracle identifier for the session.
Serial #	The serial number assigned to the session.
Status	The status of the session, ACTIVE or INACTIVE.
Machine	The name of the client machine name that the session is using.
Program	The executable the process is using against the server.
CPU	The CPU used by the process when the call started.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

CPU Tab for SQL Server

The table below describes the information available on the CPU tab of the Top Sessions view for SQL Server:

Column	Description
PID	The process ID.
User Name	The user name assigned to the process.
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
CPU	The cumulative CPU time for the process.
Pct CPU Used	The percentage of the CPU dedicated to this process.
Database	The database in which the process is running.
Command	The command the process is currently issuing.
Open Trans	The number of open transactions for the process.
Blocked	The PID of any process blocking the current process.
Wait Time	The current wait time for the process, in milliseconds.

TIP: To configure the grid to show/hide row numbers, use the Options Editor.

CPU Tab for Sybase

The table below describes the information available on the CPU tab of the Top Sessions view for Sybase:

Column	Description
PID	The process ID.
User Name	The user name assigned to the process.
FID	The process ID of the worker process' parent.

Column	Description
Status	Indicates if the process is actively performing work, is idle, blocked by another process, etc.
Host	The machine name that originated the process.
Program	The executable the process is using against the server.
CPU	The cumulative CPU time for the process in ticks.
Pct CPU Used	The percentage of the CPU dedicated to this process.
Database	The database in which the process is running.
Command	The command the process is currently issuing.
Transaction	The name of any transaction.
Blocked	The PID of any process blocking the current process.
Time Blocked	The time that the process has been blocked, in seconds.

To configure the grid to show/hide row numbers, use the Options Editor.

CPU Tab for DB2

The table below describes the information available on the CPU tab of the Top Sessions view for DB2:

Column	Description
Agent ID	The application handle of the agent holding a lock for which this application is waiting.
Auth ID	The authorization ID of the user who invoked the application that is being monitored.
OS User ID	The operating system ID of the process.
Client PID	The process ID of the client application that made the connection to the database.
Application	The name of the application executable.
Status	The current status of the application.
UOW Elapsed Time (sec)	The elapsed execution time of the most recently completed unit of work.
Agent ID	The application handle of the agent holding a lock for which this application is waiting.
User CPU Time (sec)	The total user CPU time used by the application agents.
System CPU Time (sec)	The total system CPU time used by the application agents.
Total CPU Time (sec)	The total user + system CPU time used by the application agents.

To configure the grid to show/hide row numbers, use the Options Editor.

Top SQL View

- Metrics

One or two bad queries can cause a lot of trouble for the remaining sessions in a database. It is important to find them before they get into a production system, but sometimes a few sneak through.

By applying custom filters and performance-related thresholds, the Top SQL view locates inefficient SQL. By applying filters to certain I/O and statistical counters, you hope to isolate queries that far exceed their nearest competitors in the same area (like disk reads). When you find them, you should reduce the number of sorts a query performs. Or, for a query that returns only a few records, you should try to minimize the number of rows a query processes.

The Top SQL view displays requested SQL for SQL Server, Oracle, DB2, and Sybase datasources.

Metrics

When you begin to look for inefficient SQL in a database, there are two primary questions you need to answer:

1 What has been the worst SQL that has historically been run in my database?

2 What is the worst SQL that's running right now in my database?

When troubleshooting a slow system, you should be on the lookout for any query that shows an execution count that is significantly larger than any other query on the system. It may be that the query is in an inefficient Transact SQL loop, or other problematic programming construct. Only by bringing the query to the attention of the application developers will you know if the query is being mishandled from a programming standpoint.

Index