Database Monitoring Metrics

Related page:

Metric Browser
This document describes metrics collected by the Database Agent. It shows the path to the metric from the main AppDynamics menu down to the relevant branch of the Metric Browser tree. Some of these metrics are displayed in the Database Visibility user interface and so are also documented with the name of the window, column, section, and metric name.

The Metric Browser window name uses the following format:

- Metric Browser - application_name: When you access the Metric Browser from Applications.
- Metric Browser - Database Monitoring: When you access the Metric Browser from Infrastructure.

For most types of metrics in the browser, you can click any of the points in the graph to view more information about the metric observed at that point in time. The information shown includes the metric identifier, date and time of the observation, along with any of the following values relevant to the metric:

- Obs (observed value): the average of all data points seen for that interval. For a cluster or a time rollup, this represents the weighted average across nodes or over time.
- Min: the minimum data point value seen for that interval
- Max: the maximum data point value seen for that interval
- Sum: the sum of all data point values seen for that interval.
- Count: number of observations aggregated in that one point. For example, a count of 5 indicates that there were 5 1-minute data points aggregated into one point.

Key Performance Indicators (KPI)

These metrics are available for all database platforms supported.

Calls per Minute: The number of SQL calls to the database per minute.

DB Availability: The times when the database is available (has an active connection). If the database is not available, nothing is reported.

Number of Connections: The number of connections established with the database at any point during the selected time period. A connection is a session established between a database client and a server. Since the value displayed is the average number of connections over a time range, it is common to see a very low number or 0 for this metric.

Time Spent in Execution (s): The current amount of time the database spent executing SQL statements.

Total Database Size: The amount of disk space (in MB) that the database is using.

Total Lock Time: Total time the database was in lock state in seconds.

Couchbase Metrics

Gc_num: Number of objects garbage collected
Gc_pause_time: Garbage collection pause time
Gc_pause_percent: Garbage collection pause percentage

Memory_usage: Memory (in GB) currently used by Couchbase
Memory_total: Memory (in GB) used by Couchbase over the total period of time
Memory_system: Memory (in GB) used by the system.

Cpu_user_percent: Percentage of CPU used by the user
Cpu_sys_percent: Percentage of CPU used by the system

Request_completed_count: Number of requests completed
Request_active_count: Number of active requests
Request_per_sec_1min: Requests per second in the last minute
Request_per_sec_5min: Requests per second in the last 5 minutes
Request_per_sec_15min: Requests per second in the last 15 minutes
Request_time_mean: Average request time
Request_time_median: Median request time
Request_time_80: 80th percentile request time
Request_time_95: 95th percentile request time
Request_time_99: 99th percentile request time
Request_prepared: Number of requests prepared

cpu_utilization_rate: Rate of CPU utilization
swap_total: Total amount of swap available
swap_used: Amount of swap used
mem_total: Total available memory
mem_free: Total amount of free memory
**cmd_get**: Compare and Swap gets

**couch_docs_actual_disk_size**: The physical memory used in the node

**couch_docs_data_size**: Data size of couch documents associated with a node

**couch_temporal_data_size**: Size of object data for spatial views

**couch_views_actual_disk_size**: Amount of disk space occupied by Couch views

**couch_views_data_size**: Size of object data for Couch views

**curr_items**: Number of active items in memory

**curr_items_tot**: Total number of items

**ep_bg_fetch**: Disk reads per second

**get_hits**: Number of get hits

**mem_used**: Engine's total memory usage

**ops**: Spatial operations

**vb_replica_curr_items**: Number of in-memory items

**operations per second**: The total amount of operations per second (including XDCR) to this bucket (measured from cmd_lookup + cmd_set + incr_misses + incr_hits + decr_misses + decr_hits + delete_misses + delete_hits + ep_num_ops_del_meta + ep_num_ops_get_meta + ep_num_ops_set_meta).

**gets per sec**: The number of reads (get operations) per second from this bucket (measured from cmd_get).

**sets per sec**: The number of writes (set operations) per second to this bucket (measured from cmd_set).

**deletes per sec**: The number of delete operations per second for this bucket (measured from delete_hits).

**disk creates per sec**: The number of new items created on disk per second for this bucket (measured from vb_active_ops_create + vb_replica_ops_create + vb_pending_ops_create).

**disk updates per sec**: The number of items updated on disk per second for this bucket (measured from vb_active_ops_update + vb_replica_ops_update + vb_pending_ops_update).

**disk reads per sec**: The number of reads per second from disk for this bucket (measured from ep_bg_fetches).

**disk read failures**: The number of disk read failures (measured from ep_data_read_failed).

**disk write failures**: The number of disk write failures (measured from ep_data_write_failed).

**temp OOM per sec**: The number of back-offs sent per second to client SDKs due to the out-of-memory situations from this bucket (measured from ep_temp_oom_errors).

**cache miss ratio**: The percentage of active items cached in RAM in this bucket (measured from vb_active_resident_items_ratio).

**active docs resident %**: The percentage of active items cached in RAM in this bucket (measured from vb_active_resident_items_ratio).

**DEADLOCKS**: The number of deadlocks that have occurred.

**get_hits**: Number of get hits

**mem_used**: Engine's total memory usage

**ops**: Spatial operations

**vb_replica_curr_items**: Number of in-memory items

**operations per second**: The total amount of operations per second (including XDCR) to this bucket (measured from cmd_lookup + cmd_set + incr_misses + incr_hits + decr_misses + decr_hits + delete_misses + delete_hits + ep_num_ops_del_meta + ep_num_ops_get_meta + ep_num_ops_set_meta).

**gets per sec**: The number of reads (get operations) per second from this bucket (measured from cmd_get).

**sets per sec**: The number of writes (set operations) per second to this bucket (measured from cmd_set).

**deletes per sec**: The number of delete operations per second for this bucket (measured from delete_hits).

**disk creates per sec**: The number of new items created on disk per second for this bucket (measured from vb_active_ops_create + vb_replica_ops_create + vb_pending_ops_create).

**disk updates per sec**: The number of items updated on disk per second for this bucket (measured from vb_active_ops_update + vb_replica_ops_update + vb_pending_ops_update).

**disk reads per sec**: The number of reads per second from disk for this bucket (measured from ep_bg_fetches).

**disk read failures**: The number of disk read failures (measured from ep_data_read_failed).

**disk write failures**: The number of disk write failures (measured from ep_data_write_failed).

**temp OOM per sec**: The number of back-offs sent per second to client SDKs due to the out-of-memory situations from this bucket (measured from ep_temp_oom_errors).

**cache miss ratio**: The percentage of active items cached in RAM in this bucket (measured from vb_active_resident_items_ratio).

**ACTIVE_SORTS**: The number of sorts in the database that currently have a sort heap allocated.

**AGENTS_TOP**: At the database level, it is the maximum number of agents for all applications.

**APPLS_CURCONS**: The number of applications that are currently connected to the database.

**APPLS_IN_DB2**: The number of applications that are currently connected to the database, and for which the database manager is currently processing a request.

**BINDS_PRECOMPLIES**: The number of binds and pre-compiles attempted. You can use this element to gain insight into the current level of activity within the database manager. This value does not include the count of int_auto_rebinds, but it does include binds that occur as a result of the REBIND PACKAGE command.

**COORD_AGENTS_TOP**: The highest number of coordinating agents. If the peak number of coordinating agents represents too high a workload for this node, you can reduce this upper boundary by changing the max_coordagents configuration parameter.

**DDL_SQL_STMTS**: The number of SQL Data Definition Language (DDL) statements that were executed.

**DEADLOCKS**: The number of deadlocks that have occurred.

**DIRECT_READ_REQS**: Use the following formula to calculate the average number of sectors that are read by a direct read: direct_reads / direct_read_reqs.

**DIRECT_READ_TIME**: Time spent doing direct read operations.

**DIRECT_READS**: Direct reads are performed in units, the smallest being a 512-byte sector. They are used when: Reading LONG VARCHAR columns, Reading LOB (large object) columns, Performing a backup.

**DIRECT_WRITE_REQS**: Use the following formula to calculate the average number of sectors that are written by a direct write: direct_writes / direct_write_reqs.

**DIRECT_WRITE_TIME**: Time spent doing direct write operations.

**DIRECT_WRITES**: Direct writes are performed in units, the smallest being a 512-byte sector. They are used when: Writing LONG VARCHAR columns, Writing LOB (large object) columns, Performing a restore, Performing a load, Allocating new extents for SMS table space if MPFA is enabled (which is the default).
**DYNAMIC_SQL_STMTS**: The number of dynamic SQL statements that were attempted.

**ELAPSED_EXEC_TIME_MS**: The total time (in milliseconds) required to execute all the statements for a particular application during the specified time period.

**ELAPSED_EXEC_TIME_S**: The total time (in seconds) required to execute all the statements for a particular application during the specified time period.

**FAILED_SQL_STMTS**: The number of SQL statements that were attempted, but failed.

**INT_AUTO_REBINDS**: The number of commits initiated internally by the database manager.

**INT_COMMITS**: The number of commits initiated internally by the database manager.

**INT_DEADLOCK_ROLLBACKS**: The 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.

**INT_ROLLBACKS**: The number of rollbacks initiated internally by the database manager.

**INT_ROWS_DELETED**: The number of rows deleted from the database as a result of internal activity.

**INT_ROWS_INSERTED**: The number of rows inserted into the database as a result of internal activity caused by triggers.

**INT_ROWS_UPDATED**: The number of rows updated from the database as a result of internal activity.

**LOCK_ESCALS**: The number of times that locks have been escalated from several row locks to a table lock.

**LOCK_LIST_IN_USE**: The total number of bytes of lock list memory in use.

**LOCK_TIMEOUTS**: The number of times that a request to lock an object timed-out instead of being granted.

**LOCK_WAIT_TIME**: The total elapsed time (in milliseconds) spent waiting for locks.

**LOCK_WAITS**: The number of times that applications or connections waited for locks.

**LOCKS_HELD**: The number of locks currently held.

**LOCKS_WAITING**: The number of agents waiting on a lock.

**LOG_HELD_BY_DIRTY_PAGES**: The amount of log (in bytes) corresponding to the difference between the oldest dirty page in the database and the top of the active log.

**LOG_READ_TIME_NS**: The total elapsed time spent by the logger reading log data from the disk. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

**LOG_READ_TIME_S**: At the database level, this is the number of subagents for all applications.

**LOG_READS**: The number of log pages read from disk by the logger.

**LOG_TO_REDO_FOR_RECOVERY**: The size of the log (in bytes) that will have to be redone for crash recovery.

**LOG_WRITE_TIME_NS**: The total elapsed time spent by the logger writing log data to the disk. For event monitors that write to tables, the value of this element is given in microseconds by using the BIGINT data type.

**LOG_WRITE_TIME_S**: At the database level, this is the number of subagents for all applications.

**LOG_WRITES**: The number of log pages written to disk by the logger.

**NUM_ASSOC_AGENTS**: At the database level, this is the number of subagents for all applications.

**NUM_INDOUBT_TRANS**: The number of outstanding indoubt transactions in the database. Indoubt transactions hold log space for uncommitted transactions, which can cause the logs to become full. When the logs are full, further transactions cannot be completed. The resolution of this problem involves a manual process of heuristically resolving the indoubt transactions. This monitor element provides a count of the number of currently outstanding indoubt transactions that must be heuristically resolved.

**NUM_LOG_BUFFER_FULL**: The number of times agents had to wait for log data to write to disk while copying log records into the log buffer.

**NUM_LOG_DATA_FOUND_IN_BUFFER**: The number of times log data was read from buffer instead of from disk, which is slower.

**NUM_LOG_PART_PAGE_IO**: Number of I/O requests issued by the logger for writing partial log data to disk. To determine if the current disk is adequate for logging, use this metric in conjunction with log_writes, log_write_time, and num_log_write_io.

**NUM_LOG_READ_IO**: Number of I/O requests issued by the logger to read log data from disk. To determine if the current disk is adequate for logging, use this metric in conjunction with log_reads and log_read_time.

**NUM_LOG_WRITE_IO**: Number of I/O requests issued by the logger to write log data to disk. To determine if the current disk is adequate for logging, use this metric in conjunction with log_writes and log_write_time.
POOL_ASYNC_DATA_READ_REQS: Number of asynchronous read requests by the prefetcher to the operating system. These requests are usually large block I/Os of multiple pages.

POOL_ASYNC_DATA_READS: Number of data pages read in from the table space physical containers by asynchronous engine dispatchable units for all types of table spaces. To determine the number of physical read that were performed synchronously, use this metric along with the pool_data_p_reads metrics.

POOL_ASYNC_DATA_WRITES: Number of times a buffer pool data page was physically written to disk by either an asynchronous page cleaner or prefetcher. To determine how well the buffer page cleaners are performing, use this metric in conjunction with pool_data_writes and pool_async_data_writes.

POOL_ASYNC_INDEX_READS: Number of index pages read in from the physical table space containers by asynchronous engine dispatchable units for all types of table spaces. To determine how well the prefetchers are working, compare the ratio of asynchronous reads to total physical reads. Use this information to tune the num_ioservers configuration parameter.

POOL_ASYNC_INDEX_WRITES: Number of times a buffer pool index page was physically written to disk by either an asynchronous page cleaner or prefetcher.

POOL_ASYNC_READ_TIME: Total number of milliseconds spent reading in data and index pages from physical table space containers by asynchronous engine dispatchable units for all types of table spaces. Use this metric to analyze the I/O work being performed.

POOL_ASYNC_WRITE_TIME: Total number of milliseconds spent writing data or index pages from the buffer pool to disk by database manager page cleaners. Use this metric to analyze the I/O work being performed.

POOL_DATA_FROM_ESTORE: Number of buffer pool data pages read from the extended storage monitor.

POOL_DATA_L_READS: Number of data pages which have been requested from the buffer pool (logical) for regular and large table spaces. The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

POOL_DATA_P_READS: Number of data pages read in from the table space containers (physical) for regular and large table spaces. The functionality to record buffer pool information at the statement 7 level is supported for API and CLP snapshot requests.

POOL_DATA_WRITES: Number of times a buffer pool data page was physically written to disk.

POOL_DRTY_PG_STEAL_CLNS: Number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the database.

POOL_DRTY_PGTHRSH_CLNS: Number of times a page cleaner was invoked because a buffer pool had reached the dirty page threshold criterion for the database.

POOL_INDEX_L_READS: Number of index pages which have been requested from the buffer pool (logical) for regular and large table spaces. The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

POOL_INDEX_P_READS: Number of index pages read in from the table space containers (physical) for regular and large table spaces. The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

POOL_INDEX_WRITES: Number of times a buffer pool index page was physically written to disk.

POOL_LSN_GAP_CLNS: Number of times a page cleaner was invoked because the logging space used had reached a predefined criterion for the database.

POOL_NO_VICTIM_BUFFER: Number of times an agent did not have a preselected victim buffer available.

POOL_READ_TIME: Number of milliseconds spent reading in data and index pages from the physical table space for all types of table spaces.

POOL_TEMP_DATA_L_READS: Number of data pages which have been requested from the logical buffer pool for temporary table spaces.

POOL_TEMP_DATA_P_READS: Number of data pages read in from the physical table space containers for temporary table spaces.

POOL_TEMP_INDEX_L_READS: Number of index pages which were requested from the logical buffer pool for temporary table spaces.

POOL_TEMP_INDEX_P_READS: Number of index pages read in from the physical table space containers for temporary table spaces.

POOL_WRITE_TIME: Number of milliseconds spent physically writing data or index pages from the buffer pool to disk.

PREFETCH_WAIT_TIME: Number of milliseconds spent waiting for an I/O prefetcher server to finish loading pages into the buffer pool.

ROWS_DELETED: Number of row deletions attempted.

ROWS_INSERTED: Number of row insertions attempted.

ROWS_READ: Number of rows read from tables.

ROWS_SELECTED: Number of rows that have been selected and returned to the application.

ROWS_UPDATED: Number of row updates attempted.
**SEC_LOG_USED_TOP**: The maximum number of bytes of secondary log space used.

**SEC_LOGS_ALLOCATED**: Number of secondary log files currently being used for the database.

**SELECT_SQL_STMTS**: The number of SQL SELECT statements that were executed.

**SORT_HEAP_ALLOCATED**: The number of allocated pages of sort heap space for all sorts at the level chosen and at the current time.

**SORT_OVERFLOWS**: The number of sorts that ran out of sort heap and may have required disk space for temporary storage.

**SORT_SHRHEAP_ALLOCATED**: The total amount of shared sort memory allocated in the database.

**SORT_SHRHEAP_TOP**: Database-wide shared sort memory high-water mark in 4k pages.

**STATIC_SQL_STMTS**: The number of static SQL statements that were attempted.

**TOT_LOG_USED_TOP**: The maximum number of bytes used for log space.

**TOTAL_APP_COMMITS**: Total number of commit statements issued by the client application.

**TOTAL_APP_ROLLBACKS**: Total number of rollback statements issued by the client application.

**TOTAL_CONS**: Number of newly opened connections to the database.

**TOTAL_LOG_AVAILABLE**: Number of bytes of active log space in the database that is not being used by uncommitted transactions.

**TOTAL_LOG_USED**: Number of bytes of active log space currently used in the database.

**TOTAL_SEC_CONS**: The number of connections made by a subagent to the database at the node.

**TOTAL_SORT_TIME**: The total elapsed time (in milliseconds) for all sorts that have been executed.

**TOTAL_LOCK_TIME**: The total time for which database is in locked state.

**TOTAL_SORTS**: The number of sorts that have been executed.

**TOT_LOG_USED_TOP**: The maximum amount of total log space used (in bytes).

**UID_SQL_STMTS**: Number of SQL UPDATE, INSERT, and DELETE statements that were executed.

**UNREAD_PREFETCH_PAGES**: Number of pages that the prefetcher read in that were never used.

---

**MongoDB Server Metrics**

asserts_msg: The number of message asserts. These are internal server errors that have a well defined text string. Stack traces are logged for these.

asserts_regular: The number of regular asserts raised since this process started.

asserts_user: The number of user asserts. These are errors that can be generated by a user such as out of disk space or duplicate key.

asserts_warning: The number of warnings raised since this process started.

BackgroundFlushing_flushes: The number of times the database has flushed all writes to disk.

BackgroundFlushing_total_ms: The number of milliseconds (ms) that the mongod processes have spent writing (i.e. flushing) data to disk.

Connections_available: The number of unused available connections that the database can provide. Consider this value in combination with the value of current to understand the connection load on the database, and the UNIX ulimit Settings document for more information about system thresholds on available connections.

Connections_current: The number of connections to the database server from clients. This number includes the current shell session. Consider the value of available to add more context to this datum. This figure will include the current shell connection as well as any inter-node connections to support a replica set or sharded cluster.

Cursor_timedOut: The number of cursors that have timed out since the server process started. If this number is large or growing at a regular rate, this may indicate an application error.

Cursor_totalOpen: The number of cursors that MongoDB is maintaining for clients. Because MongoDB exhausts unused cursors, typically this value is small or zero. However, if there is a queue, stale tailable cursors, or a large number of operations this value may rise.

globalLock_ActiveClients: The number of connected clients.

globalLock_CurrentQueue: The number of operations queued waiting for the lock. A consistently small queue, particularly of shorter operations should cause no concern. Also, consider this value in light of the size of queue waiting for the read lock (e.g. readers) and write lock (e.g. writers) individually.

Mem_mapped: The number of megabytes of memory mapped by the database. Because MongoDB uses memory-mapped files, this value will be roughly equivalent to the total size of your databases.
**Mem_resident**: The amount of RAM, in megabytes (MB), currently used by the database process. In normal use this value tends to grow. In dedicated database servers this number tends to approach the total amount of system memory.

**Mem_virtual**: The quantity, in megabytes (MB), of virtual memory used by the mongod process. With journaling enabled, the value of virtual is at least twice the value of mapped. If virtual value is significantly larger than mapped (e.g. 3 or more times), this may indicate a memory leak.

**Network_bytesIn**: The amount of network traffic, in bytes, received by this database. Use this value to ensure that network traffic sent to the mongod process is consistent with expectations and overall inter-application traffic.

**Network_bytesOut**: The amount of network traffic, in bytes, sent from this database. Use this value to ensure that network traffic sent by the mongod process is consistent with expectations and overall inter-application traffic.

**Network_numRequests**: The number of distinct requests that the server has received. Use this value to provide context for the bytesIn and bytesOut values to ensure that MongoDB network utilization is consistent with expectations and application use.

**OpCounters_command**: The number of commands sent to MongoDB.

**OpCounters_delete**: The number of Delete operations.

**OpCounters_getmore**: The number of GetMore operations.

**OpCounters_insert**: The number of Insert operations.

**OpCounters_query**: The number of Query operations.

**OpCounters_update**: The number of Update operations.

**Oplog_Max_Size** *(new in 4.5.4)*: The maximum size of the operation log

**Oplog_Size** *(new in 4.5.4)*: The size of the operation log

**OplogTimeDiff_in_sec** *(new in 4.5.4)*: The difference (in seconds) between the first entry in the log and the last entry in the log

**Repl_command**: The number of Replicated Commands issued to the database.

**Repl_delete**: The number of Replicated Delete operations.

**Repl_getmore**: The number of GetMore operations. This counter can be high even if the query count is low. Secondary nodes send getMore operations as part of the replication process.

**Repl_insert**: The number of replicated insert operations.

**Repl_query**: The number of Replicated Queries.

**Repl_update**: The number of Replicated Update Operations.

**Replication_MyState** *(new in 4.5.4)*: An integer between 0 and 10 that represents the replica state of the current member.

**Index_missRatio**: Ratio of index hits to misses. If there are a lot of index misses then you should look at your queries to see if they are optimally using your indexes. You may need to add new indexes to make the queries run faster. You can explain the query to see which index queries are hitting and the total execution time so you can compare query performance before and after adding the new indexes.

**Microsoft SQL Server Metrics**

**Active cursors**: The number of active cursors.

**Active Temp Tables**: The number of temporary tables/table variables in use.

**Active Transactions**: The number of active transactions for the database.

**AU cleanup batches/sec**: The number of batches per second that were completed successfully by the background task that cleans up deferred dropped allocation units.

**AU cleanups/sec**: The number of allocation units per second that were successfully dropped the background task that cleans up deferred dropped allocation units. Each allocation unit drop requires multiple batches.

**Auto-Param Attempts/sec**: The number of auto-parameterization attempts per second. Total should be the sum of the failed, safe, and unsafe auto-parameterizations. Auto-parameterization occurs when an instance of SQL Server tries to parameterize a Transact-SQL request by replacing some literals with parameters so that reuse of the resulting cached execution plan across multiple similar-looking requests is possible. Note that auto-parameterizations are also known as simple parameterizations in newer versions of SQL Server. This counter does not include forced parameterizations.

**Average Latch Wait Time (ms)**: Average latch wait time (in milliseconds) for latch requests that had to wait.

**Average Wait Time (ms)**: Average amount of wait time (in milliseconds) for each lock request that resulted in a wait.
Backup/Restore Throughput/sec: Read/write throughput for backup and restore operations of a database per second. For example, you can measure how the performance of the database backup operation changes when more backup devices are used in parallel or when faster devices are used. Throughput of a database backup or restore operation allows you to determine the progress and performance of your backup and restore operations.

Batch Requests/sec: The number of Transact-SQL command batches received per second. This statistic is affected by all constraints (such as I/O, number of users, cache size, complexity of requests, and so on). High batch requests mean good throughput.

Buffer cache hit ratio: Percentage of pages found in the buffer cache without having to read from disk. The ratio is the total number of cache hits divided by the total number of cache lookups over the last few thousand page accesses. The ratio is displayed as a percentage. After a long period of time, the ratio moves very little. Because reading from the cache is much less expensive than reading from disk, you want this ratio to be high. Generally, you can increase the buffer cache hit ratio by increasing the amount of memory available to SQL Server.

Bulk Copy Rows/sec: The number of rows bulk copied per second.

Bulk Copy Throughput/sec: Amount of data bulk copied (in kilobytes) per second.

By-reference Lob Create Count: Count of large object (lob) values that were passed by reference. By-reference lobs are used in certain bulk operations to avoid the cost of passing them by value.

By-reference Lob Use Count: Count of by-reference lob values that were used. By-reference lobs are used in certain bulk operations to avoid the cost of passing them by-value.

Cache Hit Ratio: Ratio between cache hits and lookups. The ratio is displayed as a percentage.

Cache Object Counts: The number of cache objects in the cache.

Cache Objects in use: The number of cache objects in use.

Cache Pages: The number of 8-kilobyte (KB) pages used by cache objects.

Cached Cursor Counts: The number of cursors of a given type in the cache.

Checkpoint pages/sec: The number of pages flushed to disk per second by a checkpoint or other operation that require all dirty pages to be flushed.

CLR Execution: Total execution time in CLR (microseconds)

Connection Memory (KB): Total amount of dynamic memory the server is using for maintaining connections.

Count Lob Read ahead: Count of lob pages on which readahead was issued.

Count Pull In Row: Count of column values that were pulled in-row from off-row.

Count Push Off Row: Count of column values that were pushed from in-row to off-row.

Cursor memory usage: Amount of memory consumed by cursors in kilobytes (KB).

Count/sec: Times each type of cached cursor has been used. Cursor memory usage Amount of memory consumed by cursors in kilobytes (KB).

Cursor Requests/sec: The number of SQL cursor requests received by the server. Cursor worktable usage Number of worktables used by cursors. Data File(s) Size (KB) Cumulative size (in kilobytes) of all the data files in the database including any automatic growth. Monitoring this counter is useful, for example, for determining the correct size of Database pages Number of pages in the buffer pool with database content.

Cursor worktable usage: Number of worktables in use by cursors.

Data File(s) Size(KB): Cumulative size (Kb) of all data files in the database including any automatic growth. Monitoring this counter to determine the correct size of tempdb.

Database pages: Number of database pages in use.

DBCC Logical Scan Bytes/sec: The number of logical read scan bytes per second for database console commands (DBCC).

Deferred Dropped rowsets: The number of rowsets created as a result of aborted online index build operations that are waiting to be dropped by the background task that cleans up deferred dropped rowsets.

Dropped rowset cleanups/sec: The number of rowsets per second created as a result of aborted online index build operations that were successfully dropped by the background task that cleans up deferred dropped rowsets.

Dropped rowsets skipped/sec: The number of rowsets per second created as a result of aborted online index build operations that were skipped by the background task that cleans up deferred dropped rowsets created.

Errors/sec: The number of errors/sec.

Event Notifications Delayed Drop: The number of event notifications waiting to be dropped by a system thread.

Extent Deallocations/sec: The number of extents deallocated per second in all databases in this instance of SQL Server.

Extents Allocated/sec: The number of extents allocated per second in all databases in this instance of SQL Server.
Failed AU cleanup batches/sec: The number of batches per second that failed and required retry, by the background task that cleans up deferred dropped allocation units. Failure could be due to lack of memory or disk space, hardware failure, and other reasons.

Failed Auto-Params/sec: The number of failed auto-parameterization attempts per second. This should be small. Note that auto-parameterizations are also known as simple parameterizations in later versions of SQL Server.

Failed leaf page cookie: The number of times that a leaf page cookie could not be used during an index search since changes happened on the leaf page. The cookie is used to speed up the index search.

Failed tree page cookie: The number of times that a tree page cookie could not be used during an index search since changes happened on the parent pages of those tree pages. The cookie is used to speed up the index search.

Forced Parameterizations/sec: The number of successful forced parameterizations per second.

Forwarded Records/sec: The number of records per second fetched through forwarded record pointers.

Free list stalls/sec: The number of requests per second that had to wait for a free page.

Free Space in tempdb (KB): The amount of space (in kilobytes) available in FreeSpace Page.

FreeSpace Page Fetches/sec: Number of pages fetched/second by free space scans. These scans search for free space within pages already allocated to an allocation unit, to address requests to insert or modify record fragments.

FreeSpace Scans/sec: The number of scans per second that were initiated to search for free space within pages already allocated to an allocation unit to insert or modify record fragment. Each scan may find multiple pages.

Full Scans/sec: The number of unrestricted full scans per second. These can be either base-table or full-index scans.

Granted Workspace Memory (KB): Total amount of memory currently granted to executing processes such as hash, sort, bulk copy, and index creation operations.

Guided Plan Executions/sec: The number of plan executions per second in which the query plan has been generated by using a plan guide.

HTTP Authenticated Requests: The number of authenticated HTTP requests started per second. Index Searches/sec Number of index searches per second. These are used to start a range scan, reposition a range scan, revalidate a scan point, fetch a single index record, and search down the index to locate where to insert a new row.

Index Searches/sec: Number of index searches per second. These are used to start a range scan, reposition a range scan, revalidate a scan point, fetch a single index record, and search down the index to locate where to insert a new row.

Latch Waits/sec: The number of latch requests that could not be granted immediately.

Lazy writes/sec: The number of buffers written per second by the buffer manager's lazy writer. The lazy writer is a system process that flushes out batches of dirty, aged buffers (buffers that contain changes that must be written back to disk before the buffer can be reused for a different page) and makes them available to user processes. The lazy writer eliminates the need to perform frequent checkpoints in order to create available buffers.

LobHandle Create Count: Count of temporary lobs created.

LobHandle Destroy Count: Count of temporary lobs destroyed.

LobSS Provider Create Count: Count of LOB Storage Service Providers (LobSSP) created. One worktable created per LobSSP.

LobSS Provider Destroy Count: Count of LobSSP destroyed.

LobSS Provider Truncation Count: Count of LobSSP truncated.

Lock Blocks: Current number of lock blocks in use on the server (refreshed periodically). A lock block represents an individual locked resource, such as a table, page, or row.

Lock Blocks Allocated: Current number of allocated lock blocks. At server startup, the number of allocated lock blocks plus the number of allocated lock owner blocks depends on the SQL Server

Lock Memory (KB): Total amount of dynamic memory the server is using for locks.

Lock Owner Blocks: The number of lock owner blocks currently in use on the server (refreshed periodically). A lock owner block represents the ownership of a lock on an object by an individual thread. Therefore, if three threads each have a shared (S) lock on a page, there will be three lock owner blocks.

Lock Owner Blocks Allocated: Current number of allocated lock owner blocks. At server startup, the number of allocated lock owner blocks and the number of allocated lock blocks depend on the SQL.

Lock Requests/sec: The number of new locks and lock conversions per second requested from the lock manager.

Lock Timeouts (timeout > 0)/sec: The number of lock requests per second that timed out, but excluding requests for NOWAIT locks.

Lock Timeouts/sec: The number of lock requests per second that timed out, including requests for NOWAIT locks.

Lock Wait Time (ms): Total wait time (in milliseconds) for locks in the last second.
Lock Waits/sec: The number of lock requests per second that required the caller to wait.

Log Bytes Flushed/sec: Total number of log bytes flushed.

Log Cache Hit Ratio: Percentage of log cache reads satisfied from the log cache.

Log Cache Reads/sec: Reads performed per second through the log manager cache.

Log File(s) Size (KB): Cumulative size (in kilobytes) of all the transaction log files in the database.

Log File(s) Used Size (KB): The cumulative used size of all the log files in the database.

Log Flush Wait Time: Total wait time (in milliseconds) to flush the log.

Log Flush Waits/sec: The number of commits per second waiting for the log flush.

Log Flushes/sec: The number of log flushes per second.

Log Growth(s): Total number of times the transaction log for the database has been expanded.

Log Shrinks: Total number of times the transaction log for the database has been shrunk.

Log Truncations: Total number of times the transaction log for the database has been truncated.

Logical Connections: The number of logical connections to the system.

Logins/sec: Total number of logins started per second. This does not include pooled connections.

Logouts/sec: Total number of logout operations started per second.

Longest Transaction Running Time: The length of time (in seconds) since the start of the transaction that has been active longer than any other current transaction.

Mars Deadlocks: The number of Mars Deadlocks detected.

Maximum Workspace Memory (KB): Maximum amount of memory available for executing processes such as hash, sort, bulk copy, and index creation operations.

Memory Grants Outstanding: Total number of processes that have successfully acquired a workspace memory grant.

Memory Grants Pending: Total number of processes waiting for a workspace memory grant.

Mixed page allocations/sec: The number of pages allocated per second from mixed extents. These could be used for storing the IAM pages and the first eight pages that are allocated to an allocation unit.

Non-atomic yield rate: The number of non-atomic yields per second.

NonSnapshot Version Transactions: The number of currently active transactions that are not using snapshot isolation level and have made data modifications that have generated row versions in the Number of active cursor plans.

Number of active cursor plans: Number of cursor plans.

Number of Deadlocks/sec: The number of lock requests per second that resulted in a deadlock.

Number of SuperLatches: The number of latches that are currently SuperLatches.

Optimizer Memory (KB): Total amount of dynamic memory the server is using for query optimization.

Page compression attempts/sec: The number of pages evaluated for page-level compression. Includes pages that were not compressed because significant savings could be achieved. Includes all objects in the instance of SQL Server. For information about specific objects, see sys.dm_db_index_operational_stats (Transact-SQL).

Page Deallocations/sec: The number of pages deallocated per second in all databases in this instance of SQL Server. These include pages from mixed extents and uniform extents.

Page life expectancy: The number of seconds a page will stay in the buffer pool without references.

Page lookups/sec: The number of requests per second to find a page in the buffer pool.

Page reads/sec: The number of physical database page reads that are issued per second. This statistic displays the total number of physical page reads across all databases. Because physical I/O is expensive, you may be able to minimize the cost, either by using a larger data cache, intelligent indexes, and more efficient queries or by changing the database design.

Page Splits/sec: The number of page splits per second that occur as the result of overflowing index pages.

Page writes/sec: The number of physical database page writes issued per second.

Pages Allocated/sec: The number of pages allocated per second in all databases in this instance of SQL Server. These include pages allocations from both mixed extents and uniform extents.
Pages compressed/sec: The number of data pages that are compressed by using PAGE compression. Includes all objects in the instance of SQL Server. For information about specific objects, see sys.dm_db_index_operational_stats (Transact-SQL).

Percent Log Used: Percentage of space in the log that is in use.

Probe Scans/sec: The number of probe scans per second that are used to find at most one single qualified row in an index or base table directly.

Processes blocked: The number of currently blocked processes.

Range Scans/sec: The number of qualified range scans through indexes per second.

Readahead pages/sec: The number of pages read per second in anticipation of use.

Repl. Trans. Rate: The number of transactions per second read out of the transaction log of the publication database and delivered to the distribution database.

Safe Auto-Params/sec: The number of safe auto-parameterization attempts per second. Safe refers to a determination that a cached execution plan can be shared between different similar-looking Transact-SQL statements. SQL Server makes many auto-parameterization attempts some of which turn out to be safe and others fail. Note that auto-parameterizations are also known as simple parameterizations in later versions of SQL Server. This does not include forced parameterizations.

Scan Point Revalidations/sec: The number of times per second that the scan point had to be revalidated to continue the scan.

Skipped Ghosted Records/sec: The number of ghosted records per second skipped during scans.

SOAP Empty Requests: The number of SOAP method invocations passed to the stored procedure (or template) with an empty string as its value (not a NULL value) in order to provide an included input parameter with no value assigned to it.

SOAP Method Invocations: The number of SOAP method invocations started per second.

SOAP Session Initiate Requests: The number of SOAP Session initiate requests started per second.

SOAP Session Terminate Requests: The number of SOAP Session terminate requests started per second.

SOAP SQL Requests: The number of SOAP SQL requests started per second.

SOAP WSDL Requests: The number of SOAP Web Service Description Language requests started per second.

SQL Attention rate: The number of attentions per second. An attention is a request by the client to end the currently running request.

SQL Cache Memory (KB): Total amount of dynamic memory the server is using for the dynamic SQL cache.

SQL Compilations/sec: The number of SQL compilations per second. Indicates the number of times the compile code path is entered. Includes compiles caused by statement-level recompilations in SQL Server. After SQL Server user activity is stable, this value reaches a steady state.

SQL Re-Compilations/sec: The number of statement recompiles per second. Counts the number of times statement recompiles are triggered. Generally, you want the recompiles to be low. In later versions of SQL Server, recompilations are statement-scoped instead of batch-scoped recompilations in Microsoft SQL Server 2000. Therefore, direct comparison of values of this counter between SQL Server and earlier versions is not possible.

Stored Procedures Invoked/sec: This counter reports the total number of activation stored procedures invoked by all queue monitors in the instance per second.

SuperLatch Demotions/sec: The number of SuperLatches that have been demoted to regular latches in the last second.

SuperLatch Promotions/sec: The number of latches that have been promoted to SuperLatches in the last second.

Table Lock Escalations/sec: The number of times locks on a table were escalated to the TABLE or HoBT granularity. Target pages Ideal number of pages in the buffer pool.

Target pages: Ideal number of pages in the buffer pool.

Target Server Memory (KB): Total amount of dynamic memory the server can consume.

Task Limit Reached: The number of times that a queue monitor would have started a new task, but did not because the maximum number of tasks for the queue is already running.

Task Limit Reached/sec: The number of times per second that a queue monitor would have started a new task, but did not because the maximum number of tasks for the queue is already running.

Tasks Aborted/sec: The number of activation stored procedure tasks that end with an error or are aborted by a queue monitor for failing to receive messages.

Tasks Running: The number of activation stored procedures that are currently running.

Tasks Started/sec: The number of activation stored procedures started per second by all queue monitors in the instance.
Temp Tables Creation Rate: The number of temporary tables/table variables created per second.

Temp Tables For Destruction: The number of temporary tables/table variables waiting to be destroyed by the cleanup system thread.

Total Latch Wait Time (ms): Total latch wait time (in milliseconds) for latch requests in the last second.

TOTAL_LOCK_TIME: The total time for which database is in locked state.

Total Server Memory (KB): The committed memory from the buffer pool (in kilobytes).

Transactions/sec: The number of transactions started for the database per second.

Unsafe Auto-Params/sec: The number of unsafe auto-parameterization attempts per second. For example, the query has some characteristics that prevent the cached plan from being shared. These are designated as unsafe. This does not count the number of forced parameterizations.

Update conflict ratio: The percentage of those transactions using the snapshot isolation level that have encountered update conflicts within the last second. An update conflict occurs when a snapshot isolation level transaction attempts to modify a row that last was modified by another transaction that was not committed when the snapshot isolation level transaction started.

Update Snapshot Transactions: The number of currently active transactions using the snapshot isolation level and have modified data.

Used leaf page cookie: The number of times a leaf page cookie is used successfully during an index search since no change happened on the leaf page. The cookie is used to speed up the index search.

Used tree page cookie: The number of times a tree page cookie is used successfully during an index search since no change happened on the parent page of the tree page. The cookie is used to speed up the index search.

User Connections: The number of users currently connected to SQL Server.

Version Cleanup rate (KB/s): The rate (in kilobytes per second) at which row versions are removed from the snapshot isolation version store.

Version Generation rate (KB/s): The rate (in kilobytes per second) at which new row versions are added to the snapshot isolation version store.

Version Store Size (KB): The amount of space (in kilobytes) in Version Store unit count. The number of active allocation units in the snapshot isolation version store.

Version Store unit creation: The number of allocation units that have been created in the snapshot isolation store since the instance of the Database Engine was started.

Version Store unit truncation: The number of allocation units that have been removed from the snapshot isolation store since the instance of the Database Engine was started.

Workfiles Created/sec: The number of work files created per second. For example, work files could be used to store temporary results for hash joins and hash aggregates.

Worktables Created/sec: The number of work tables created per second. For example, work tables could be used to store temporary results for query spool, lob variables, XML variables, and cursors.

Worktables From Cache Ratio: Percentage of work tables created where the initial two pages of the work table were not allocated but were immediately available from the work table cache. (When a work table is dropped, two pages may remain allocated and they are returned to the work table cache. This increases performance.)

Write Transactions/sec: The number of transactions that wrote to the database and committed, in the last second.

Microsoft SQL Azure Metrics

avg_cpu_percent: Average compute utilization in the percentage of the limit of the service tier.

avg_data_io_percent: Average data I/O utilization in percentage based on the limit of the service tier.

avg_log_write_percent: Average write resource utilization in the percentage of the limit of the service tier.

avg_memory_percent: Average memory utilization in the percentage of the limit of the service tier.

MySQL Server Metrics

Aborted_clients: The number of clients that were aborted (because they did not properly close the connection to the MySQL server). For some applications, this can be OK, but for some other applications you might want to track the value, as aborted connects may indicate some sort of application failure.

Aborted_connects: The number of failed attempts to connect to the MySQL server.

Bytes_received: The number of bytes received from all clients.

Bytes_sent: The number of bytes sent to all clients.
Com_alter_table: The number of times each ALTERTABLE statement has been executed.

Com_create_index: The number of times each CREATE INDEX statement has been executed.

Com_create_table: The number of times each CREATE TABLE statement has been executed.

Com_delete: The number of times each DELETE statement has been executed.

Com_insert: The number of times each INSERT statement has been executed.

Com_optimize: The number of times each OPTIMIZE statement has been executed.

Com_select: The number of times each SELECT statement has been executed.

Com_update: The number of times each UPDATE statement has been executed.

Connections: The number of connection attempts (successful or not) to the MySQL server.

Created_tmp_disk_tables: The number of temporary tables on disk created automatically by the server while executing statements.

Created_tmp_files: How many temporary files mysqld has created.

Created_tmp_tables: The number of in-memory temporary tables created automatically by the server while executing statements. If Created_tmp_disk_tables is large, you may want to increase the tmp_table_size value to cause temporary tables to be memory-based instead of disk-based.

Handler_delete: The number of times that rows have been deleted from tables.

Innodb_buffer_pool_pages_data: The number of pages containing data (dirty or clean).

Innodb_buffer_pool_pages_dirty: The number of pages currently dirty.

Innodb_buffer_pool_pages_flushed: The number of buffer pool page-flush requests.

Innodb_buffer_pool_pages_free: The number of free pages.

Innodb_buffer_pool_pages_misc: The number of pages that are busy because they have been allocated for administrative overhead such as row locks or the adaptive hash index. This value can also be calculated as Innodb_buffer_pool_pages_total.

Innodb_buffer_pool_pages_total: The total size of the buffer pool, in pages.

Innodb_buffer_pool_read_ahead_rnd: The number of random read-aheads initiated by InnoDB. This happens when a query scans a large portion of a table but in random order.

Innodb_buffer_pool_read_requests: The number of logical read requests InnoDB has done.

Innodb_buffer_pool_reads: The number of logical reads that InnoDB could not satisfy from the buffer pool and had to do a single-page read.

Innodb_buffer_pool_wait_free: Normally, writes to the InnoDB buffer pool happen in the background. However, if it is necessary to read or create a page and no clean pages are available, it is also necessary to wait for pages to be flushed first. This counter counts instances of these waits. If the buffer pool size has been set properly, this value should be small.

Innodb_buffer_pool_write_requests: The number writes done to the InnoDB buffer pool.

Innodb_data_fsyncs: The number of fsync() operations so far.

Innodb_data_pending_fsyncs: The current number of pending fsync() operations.

Innodb_data_pending_reads: The current number of pending reads.

Innodb_data_pending_writes: The current number of pending writes.

Innodb_data_read: The amount of data read so far, in bytes.

Innodb_data_reads: The total number of data reads.

Innodb_data_writes: The total number of data writes.

Innodb_data_written: The amount of data written so far, in bytes.

Innodb_dbwr_pages_written: The number of doublewrite operations that have been performed.

Innodb_dbwr_writes: The number of pages that have been written for doublewrite operations.

Innodb_log_waits: The number of times that the log buffer was too small and a wait was required for it to be flushed before continuing.

Innodb_log_write_requests: The number of log write requests.
**Innodb_log_writes**: The number of physical writes to the log file.

**Innodb_pages_created**: The number of pages created.

**Innodb_pages_read**: The number of pages read.

**Innodb_pages_written**: The number of pages written.

**Innodb_row_lock_current_waits**: The number of row locks currently being waited for.

**Innodb_row_lock_time**: The total time spent in acquiring row locks, in milliseconds.

**Innodb_row_lock_time_avg**: The average time to acquire a row lock, in milliseconds.

**Innodb_row_lock_time_max**: The maximum time to acquire a row lock, in milliseconds.

**Innodb_row_lock_waits**: The number of times a row lock had to be waited for.

**Innodb_rows_deleted**: The number of rows deleted from InnoDB tables.

**Innodb_rows_inserted**: The number of rows inserted into InnoDB tables.

**Innodb_rows_read**: The number of rows read from InnoDB tables.

**Innodb_rows_updated**: The number of rows updated in InnoDB tables.

**Key_blocks_used**: The number of used blocks in the key cache. This value is a high-water mark that indicates the maximum number of blocks that have ever been in use at one time.

**Key_read_requests**: The number of requests to read a key block from the cache. **Key writes** The number of physical writes of a key block to disk.

**Key_reads**: The number of physical reads of a key block from disk. If **Key_reads** is large, then your **key_buffer_size** value is probably too small. The cache miss rate can be calculated as **Key_reads/Key_read_requests**.

**Key_write_requests**: The number of requests to write a key block to the cache.

**Key_writes**: The number of physical writes of a key block from the MyISAM key cache to the physical disk.

**Open_files**: The number of files that are open. **Open_streams** The number of streams that are open (used mainly for logging).

**Open_tables**: The number of table cache misses. If the value is large, you probably need to increase table_cache. Typically you would want this to be less than 1 or 2 opened tables per second.

**Opened_tables**: The number of tables that have been opened. The number of tables that have been opened. If **Opened_tables** is big, your **table_cache** value is probably too small.

**Qcache_free_blocks**: The number of free memory blocks in the query cache.

**Qcache_free_memory**: The amount of free memory for the query cache.

**Qcache_hits**: The number of query cache hits.

**Qcache_inserts**: The number of queries added to the query cache.

**Qcache_lowmem_prunes**: The number of queries that were deleted from the query cache because of low memory.

**Qcache_not_cached**: The number of non-cached queries (not cacheable, or not cached due to the query_cache_type setting).

**Qcache_queries_in_cache**: The number of queries registered in the query cache.

**Qcache_total_blocks**: The total number of blocks in the query cache.

**Questions**: The number of statements that clients have sent to the server.

**Seconds_Behind_Master**: The number of seconds that the replica SQL thread is behind processing the primary binary log. This field directly drives **MTTR**. A high or increased number of **Seconds_Behind_Master** indicates a slow disk or poorly tuned machine running the database. If this value reaches 45 minutes or more, the ability to recover from a crash, or after a manual failover, is delayed. If this value is tracked by the Machine Agent through the MySQL extension and shows a generally positive slope, it indicates a death spiral.

**Select_full_join**: Joins performed without keys. This should be zero. This is a good way to catch development errors, as just a few such queries can decrease the system's performance.

**Select_full_range_join**: The number of joins that used a range search on a reference table.

**Select_range**: The number of joins that used ranges on the first table. This is normally not a critical issue even if the value is quite large.

**Select_range_check**: The number of joins without keys that check for key usage after each row. If this is not 0, you should carefully check the indexes of your tables.
Select_scan: Number of queries that performed a full table scan. In some cases these are OK but their ratio to all queries should be constant. If you have the value growing it can be a problem with the optimizer, lack of indexes or some other problem.

Slave_IO_Running: Whether the I/O thread for reading the primary's binary log is running.

Slave_open_temp_tables: The number of temporary tables that the replica SQL thread currently has open.

Slave_SQL_Running: Whether the SQL thread for executing events in the relay log is running.

Slow_launch_threads: The number of threads that have taken more than slow_launch_time seconds to create.

Slow_queries: The number of queries longer than --long-query-time or that are not using indexes. These should be a small fraction of all queries. If it grows, the system will have performance problems.

Sort_merge_passes: The number of merge passes that the sort algorithm has had to do. If this value is large, you should consider increasing the value of the sort_buffer_size system variable.

Sort_range: The number of sorts that were done using ranges.

SQL_Delay: The number of seconds that the replica lags behind the primary

Threads_cached: The number of threads in the thread cache.

Threads_connected: The number of currently open connections.

Threads_created: This should be low. Higher values may mean that you need to increase the value of thread_cache or you have the number of connections increasing, which also indicates a potential problem.

Threads_running: The number of threads that are not sleeping.

Oracle Server Metrics

bytes received via SQL*Net from client: Total number of bytes received from the client over Oracle Net Services

bytes received via SQL*Net from dblink: Total number of bytes received from a database link over Oracle Net Services

bytes sent via SQL*Net to client: Total number of bytes sent to the client from the foreground processes

bytes sent via SQL*Net to dblink: Total number of bytes sent over a database link

CPU used by this session: Amount of CPU time (in 10s of milliseconds) used by a session from the time a user call starts until it ends. If a user call completes within 10 milliseconds, the start and end user-call time are the same for purposes of this statistics, and 0 milliseconds are added.

db block changes: Closely related to consistent changes, this statistic counts the total number of changes that were part of an update or delete operation that were made to all blocks in the SGA. Such changes generate redo log entries and hence become permanent changes to the database if the transaction is committed. This approximates total database work. It statistic indicates the rate at which buffers are being dirtied (on a per-transaction or per-second basis, for example).

db block gets: The number of times a CURRENT block was requested. See Also: consistent gets

db block gets from cache: The number of times a CURRENT block was requested from the buffer cache. This is a subset of db block gets statistics value.

DBWR checkpoint buffers written: The number of buffers that were written for checkpoints

DBWR checkpoints: The number of times the DBWR was asked to scan the cache and write all blocks marked for a checkpoint or the end of recovery. This statistic is always larger than background checkpoints completed.

DML statements parallelized: The number of DML statements that were executed in parallel.

enqueue conversions: Total number of conversions of the state of table or row lock.

enqueue deadlocks: Total number of deadlocks between table or row locks in different sessions.

enqueue releases: Total number of table or row locks released.

enqueue requests: Total number of table or row locks acquired.

enqueue timeouts: Total number of table and row locks (acquired and converted) that timed out before they could complete.

enqueue waits: Total number of waits that occurred during an enqueue convert or get because the enqueue get was deferred

exchange deadlocks: The number of times that a process detected a potential deadlock when exchanging two buffers and raised an internal, restartable error. Index scans are the only operations that perform exchanges.

execute count: Total number of calls (user and recursive) that executed SQL statements.
**gc current block receive time:** The total time required for consistent read requests to complete. It records the round-trip time for all requests for consistent read blocks.

**index fast full scans (direct read):** The number of fast full scans initiated using direct read.

**index fast full scans (full):** The number of fast full scans initiated for full segments.

**index fast full scans (rowid ranges):** The number of fast full scans initiated with rowid endpoints specified.

**lob reads:** The number of LOB API read operations performed in the session/system. A single LOB API read may correspond to multiple physical/logical disk block reads.

**lob writes:** The number of LOB API write operations performed in the session/system. A single LOB API write may correspond to multiple physical/logical disk block writes.

**logons current:** Total number of current logons. Useful only in V$SYSSTAT.

**Number of Small Reads:** The total number of physical reads less the number of physical read total multi block requests.

**Number of Small Writes:** The total number of physical writes less the number of physical write total multi block requests.

**opened cursors current:** Total number of current open cursors.

**Parallel operations not downgraded:** The number of times parallel execution was executed at the requested degree of parallelism.

**parse count (hard):** Total number of parse calls (real parses). A hard parse is a very expensive operation in terms of memory use, because it requires Oracle to allocate a workheap and other memory structures and then build a parse tree.

**parse count (total):** Total number of parse calls (hard and soft). A soft parse is a check on an object already in the shared pool, to verify that the permissions on the underlying object have not changed.

**parse time cpu:** Total CPU time used for parsing (hard and soft) in 10s of milliseconds.

**parse time elapsed:** Total elapsed time for parsing, in 10s of milliseconds. Subtract parse time cpu from this statistic to determine the total waiting time for parse resources.

**physical read bytes:** Total size in bytes of all disk reads by application activity (and not other instance activity) only.

**physical read IO requests:** The number of read requests for application activity (mainly buffer cache and direct load operation) which read one or more database blocks per request. This is a subset of physical read total IO requests statistic.

**physical read total bytes:** Total size in bytes of disk reads by all database instance activity including application reads, backup and recovery, and other utilities. The difference between this value and physical read bytes gives the total read size in bytes by non-application workload.

**physical read total IO requests:** The number of read requests which read one or more database blocks for all instance activity including application, backup and recovery, and other utilities. The difference between this value and physical read total multi block requests gives the total number of single block read requests.

**physical reads:** Total number of data blocks read from disk. This value can be greater than the value of physical reads direct plus physical reads cache as reads into process private buffers also included in this statistic.

**physical write bytes:** Total size in bytes of all disk writes from the database application activity (and not other kinds of instance activity).

**physical write IO requests:** The number of write requests for application activity (mainly buffer cache and direct load operation) which wrote one or more database blocks per request.

**physical write total bytes:** Total size in bytes of all disk writes for the database instance including application activity, backup and recovery, and other utilities. The difference between this value and physical write bytes gives the total write size in bytes by non-application workload.

**physical write total IO requests:** The number of write requests which wrote one or more database blocks from all instance activity including application activity, backup and recovery, and other utilities. The difference between this stat and physical write total multi block requests gives the number of single block write requests.

**physical writes:** Total number of data blocks written to disk. This statistics value equals the sum of physical writes direct and physical writes from cache values.

**physical writes direct:** The number of writes directly to disk, bypassing the buffer cache (as in a direct load operation).

**physical writes from cache:** Total number of data blocks written to disk from the buffer cache. This is a subset of physical writes statistic.

**process last non-idle time:** The last time this process executed.

**queries parallelized:** The number of SELECT statements executed in parallel.

**Read Percent:**

**recovery blocks read:** The number of blocks read during recovery.
recursive calls: The number of recursive calls generated at both the user and system level. Oracle maintains tables used for internal processing. When Oracle needs to make a change to these tables, it internally generates an internal SQL statement, which in turn generates a recursive call.

recursive cpu usage: Total CPU time used by non-user calls (recursive calls). Subtract this value from CPU used by this session to determine how much CPU time was used by the user calls.

redo blocks written: This is the total number of redo blocks written. This statistic divided by "redo writes" equals the number of blocks per write.

redo buffer allocation retries: Total number of retries necessary to allocate space in the redo buffer. Retries are needed either because the redo writer has fallen behind or because an event such as a log switch is occurring.

redo log space requests: The number of times the active log file is full and Oracle must wait for disk space to be allocated for the redo log entries. Such space is created by performing a log switch.

redo log space wait time: Total elapsed waiting time for redo log space requests in 10s of milliseconds.

redo size: Total amount of redo generated in bytes.

redo synch time: Elapsed time of all redo synch writes calls in 10s of milliseconds

redo write time: Total elapsed time of the write from the redo log buffer to the current redo log file in microseconds.

redo writes: Total number of writes by LGWR to the redo log files. redo blocks written divided by this statistic equals the number of blocks per write.

session cursor cache count: Total number of cursors cached. This statistic is incremented only if SESSION_CACHED_CURSORS > 0. This statistic is the most useful in V$SESSTAT. If the value for this statistic in V$SESSTAT is close to the setting of the SESSION_CACHED_CURSORS parameter, the value of the parameter should be increased.

session cursor cache hits: The number of hits in the session cursor cache. A hit means that the SQL statement did not have to be reparsed. Subtract this statistic from parse count (total) to determine the real number of parses that occurred.

session logical reads: The sum of db block gets plus consistent gets. This includes logical reads of database blocks from either the buffer cache or process private memory.

session stored procedure space: Amount of memory this session is using for stored procedures.

Small IO Percent:

sorts (disk): The number of sort operations that required at least one disk write.

sorts (memory): The number of sort operations that were performed completely in memory and did not require any disk writes.

sorts (rows): Total number of rows sorted.

table scan rows gotten: The number of rows that are processed during scanning operations.

table scans (direct read): The number of table scans performed with direct read (bypassing the buffer cache).

table scans (long tables): Long (or conversely short) tables can be defined as tables that do not meet the short table criteria as described in table scans (short tables).

table scans (rowid ranges): During a parallel query, the number of table scans conducted with specified ROWID ranges.

transaction lock background get time: Useful only for internal debugging purposes.

transaction lock foreground wait time: Useful only for internal debugging purposes.

transaction rollbacks: The number of transactions being successfully rolled back.

TOTAL_LOCK_TIME: The total time for which database is in locked state.

user calls: The number of user calls such as login, parse, fetch, or execute

user commits: The number of user commits. When a user commits a transaction, the redo generated that reflects the changes made to database blocks must be written to disk. Commits often represent the closest thing to a user transaction rate.

user rollbacks: The number of times users manually issue the ROLLBACK statement or an error occurs during a user's transactions.

Some of the obsolete metrics were removed and the following metrics are included with 20.11 and later versions of Controller.

DB|Server Statistic|DB time: The sum of CPU consumption of all the Oracle process and the sum of non-idle wait time

DB|Server Statistic|OS System time used: The total CPU time used for system calls
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
<tr>
<td>**DB</td>
<td>Server Statistic</td>
</tr>
</tbody>
</table>

**PostgreSQL Server Metrics**

- **blks_hit**: Number of times disk blocks were found already in the buffer cache, so that a physical disk read was not necessary. This only includes hits in the PostgreSQL buffer cache, does not include the operating system's file system .cache).
- **blks_read**: Number of disk blocks read from the database.
- **confi_bufferpin**: Number of queries in the database that were canceled because of pinned buffers.
- **confi_deadlock**: Number of queries in the database that were canceled because of deadlocks
- **confi_lock**: Number of queries in the database that were canceled due because of timeouts.
- **confi_snapshot**: Number of queries in the database that were canceled because of old snapshots.
- **confi_tablespace**: Number of queries in the database that were canceled because of dropped tablespaces.
- **numbackends**: Number of backends currently connected to the database.
- **size_mb**:
- **tup_deleted**: Number of rows deleted by queries in the database.
- **tup_fetched**: Number of rows fetched by queries in the database.
- **tup_inserted**: Number of rows inserted by queries in the database.
- **tup_returned**: Number of rows returned by queries in the database.
- **tup_updated**: Number of rows updated by queries in the database.
- **xact_commit**: Number of transactions in the database that have been committed.
- **xact_rollback**: Number of transactions in the database that have been rolled back.

**Sybase ASE and IQ Server Metrics**

- **active_connections**: The number of users currently connected to the database
- **ActiveReq**: Returns the number of server threads that are currently handling a request.
- **ActiveVersionsCount**: Count of Active Transaction Versions.
- **ActiveVersionsCreateMB**: Size in Mb of versions for active transaction.
- **ActiveVersionsDeleteMB**: Size in Mb of versions for active transactions.
- **AlOs_delayed_due_to_engine_limit**:
- **AlOs_delayed_due_to_os_limit**:
- **AlOs_delayed_due_to_server_limit**:
- **BytesReceived**: Returns the number of bytes received during client/server communications. This value is updated for HTTP and HTTPS connections.
BytesReceivedUncomp: Returns the number of bytes that would have been received during client/server communications if compression was disabled. (This value is the same as the value for BytesReceived if compression is disabled.)

BytesSent: Returns the number of bytes sent during client/server communications. This value is updated for HTTP and HTTPS connections.

BytesSentUncomp: Returns the number of bytes that would have been sent during client/server communications if compression was disabled. (This value is the same as the value for BytesSent if compression is disabled.)

CacheFileDirty: Returns the number of cache pages that are dirty (needing a write).

CacheFree: Returns the number of cache pages not being used.

CacheHits: Returns the number of database page lookups.

CachePanics: Returns the number of times the cache manager has failed to find a page to allocate.

CachePinned: Returns the number of pinned cache pages.

CacheRead: Returns the number of cache reads.

CacheReplacements: Returns the number of pages in the cache that have been replaced.

CacheScavenges: Returns the number of times the cache manager has scavenged for a page to allocate.

DiskRead: Returns the number of disk reads.

mempages_allocated: The number of memory pages that have been allocated.

mempages_freed: The number of memory pages that have been de-allocated.

OtherVersionsCount: Shows count of other db versions. These versions will eventually be dropped when they are no longer referenced or referencable by active transactions.

OtherVersionsMB: Shows space usage in MB of other db versions. These versions will eventually be dropped when they are no longer referenced or referencable by active transactions.

ProcessCPU: Returns CPU usage for the database server process. Values are in seconds. This property is supported on Windows and Unix. This property is not supported on Windows Mobile.

ProcessCPUUser: Returns user CPU usage for the database server process. Values are in seconds. This excludes the amount of CPU time that the database server spent inside the operating system kernel. This property is supported on Windows and Unix. This property is not supported on Windows Mobile.

RequestsReceived: Requests received by the server.

TempBufferCapacityCount: Capacity count of Temporary Buffers.

TempBufferCapacityMB: Capacity in Mb of Temporary Buffers.

TempBufferLockedCount: The number of Temporary Buffers that are locked.

TempBufferUsedCount: The number of Temporary Buffers in use.

total_bytes_received: The number of bytes received during client/server communications. This value is updated for HTTP and HTTPS connections.

total_bytes_sent: Returns the number of bytes sent during client/server communications. This value is updated for HTTP and HTTPS connections.

TOTAL_LOCK_TIME: The total time for which database is in locked state.

xacts: The number of transactions.

yields: Returns the number of times the Adaptive Server engine(s) yielded to the operating system.