5

Measuring Database Performance

This chapter describes how to measure the performance of Oracle Database using database statistics.

This chapter contains the following topics:

- About Database Statistics
- Interpreting Database Statistics

About Database Statistics

Database statistics provide information about the type of database load and the resources being used by the database. To effectively measure database performance, statistics must be available.

Oracle Database generates many types of cumulative statistics for the system, sessions, segments, services, and individual SQL statements. Cumulative values for statistics are generally accessible using dynamic performance views, or `V$` views. When analyzing database performance in any of these scopes, look at the change in statistics (delta value) over the period you are interested in. Specifically, focus on the difference between the cumulative values of a statistic at the start and the end of the period.

This section describes some of the more important database statistics that are used to measure the performance of Oracle Database:

- Time Model Statistics
- Active Session History Statistics
- Wait Events Statistics
- Session and System Statistics

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* for information about optimizer statistics

Time Model Statistics

Time model statistics use time to identify quantitative effects about specific actions performed on the database, such as logon operations and parsing. The most important time model statistic is database time, or DB time. This statistic represents the total time spent in database calls for foreground sessions and is an indicator of the total instance workload. DB time is measured cumulatively from the time of instance startup and is calculated by aggregating the CPU and wait times of all foreground sessions not waiting on idle wait events (non-idle user sessions).

> **Note:**
>
> Because DB time is calculated by combining the times from all non-idle user foreground sessions, it is possible that the DB time can exceed the actual time elapsed after the instance started. For example, an instance that has been running for 30 minutes could have four active user sessions whose cumulative DB time is approximately 120 minutes.

When tuning an Oracle database, each component has its own set of statistics. To look at the system as a whole, it is necessary to have a common scale for comparisons. Many Oracle Database advisors and reports thus describe statistics in terms of time.

Ultimately, the objective in tuning an Oracle database is to reduce the time that users spend in performing an action on the database, or to simply reduce `DB time`. Time model statistics are accessible from the `V$SESS_TIME_MODEL` and `V$SYS_TIME_MODEL` views.

> **See Also:**
>
> *Oracle Database Reference* for information about the `V$SESS_TIME_MODEL` and `V$SYS_TIME_MODEL` views

Active Session History Statistics

An active session is any session that is either connected to the database or on a CPU, and is waiting for an event that does not belong to the Idle wait class. Oracle Database samples active sessions every second and stores the sampled data in a circular buffer in the shared global area (SGA).

The sampled session activity is accessible using the `V$ACTIVE_SESSION_HISTORY` view. Each session sample contains a set of rows and the `V$ACTIVE_SESSION_HISTORY` view returns one row for each active session per sample, starting with the latest session sample rows. Because the active session samples are stored in a circular buffer in the SGA, the greater the system activity, the smaller the number of seconds of session activity that can be stored. This means that the duration for which a session sample is displayed in the `V$` view is completely dependent on the level of database activity. Because the content of the `V$` view can become quite large during heavy system activity, only a portion of the session samples is written to disk.

By capturing only active sessions, a manageable set of data can be captured with its size being directly related to the work being performed, rather than the number of sessions allowed on the system. Active Session History (ASH) enables you to examine and perform detailed analysis on both current data in the `V$ACTIVE_SESSION_HISTORY` view and historical data in the `DBA_HIST_ACTIVE_SESS_HISTORY` view, often avoiding the need to replay the workload to trace additional performance information. ASH also contains execution plan information for each captured SQL statement. You can use this information to identify which part of SQL execution contributed most to the SQL elapsed time. The data present in ASH can be rolled up in various dimensions that it captures, including:

- SQL identifier of SQL statement
- SQL plan identifier and hash value of the SQL plan used to execute the SQL statement
- SQL execution plan information

- Object number, file number, and block number
- Wait event identifier and parameters
- Session identifier and session serial number
- Module and action name
- Client identifier of the session
- Service hash identifier
- Consumer group identifier

You can gather this information over a specified duration into an ASH report.

Active session history sampling is also available for Active Data Guard physical standby instances and Oracle Automatic Storage Management (Oracle ASM) instances. On these instances, the current session activity is collected and displayed in the `V$ACTIVE_SESSION_HISTORY` view, but not written to disk.

> **See Also:**
>
> - "Analyzing Sampled Data " for information about ASH reports
> - *Oracle Data Guard Concepts and Administration* for information about Active Data Guard physical standby databases
> - *Oracle Automatic Storage Management Administrator's Guide* for information about Oracle ASM instances

Wait Events Statistics

Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before processing could continue. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention.

To enable easier high-level analysis of wait events, Oracle Database groups events into the following classes:

- Administrative
- Application
- Cluster
- Commit
- Concurrency
- Configuration
- Idle
- Network
- Other
- Scheduler
- System I/O
- User I/O

The wait classes are based on a common solution that usually applies to fixing a problem with the particular wait event. For example, exclusive TX locks are generally an application-level issue and HW locks are generally a configuration issue. The following list includes common examples of wait events in some of the wait classes:

- Application: locks waits caused by row level locking or explicit lock commands
- Commit: waits for redo log write confirmation after a commit
- Idle: wait events that signify the session is inactive, such as `SQL*Net message from client`
- Network: waits for data to be sent over the network
- User I/O: wait for blocks to be read off a disk

Wait event statistics for a database instance include statistics for both background and foreground processes. Because tuning is typically focused in foreground activities, overall database instance activity is categorized into foreground and background statistics in the relevant `V$` views to facilitate tuning.

The `V$SYSTEM_EVENT` view shows wait event statistics for the foreground activities of a database instance and the wait event statistics for the database instance. The `V$SYSTEM_WAIT_CLASS` view shows these foreground and wait event statistics at the instance level after aggregating to wait classes. `V$SESSION_EVENT` and `V$SESSION_WAIT_CLASS` show wait event and wait class statistics at the session level.

> **See Also:**
>
> *Oracle Database Reference* for information about wait events

Session and System Statistics

A large number of cumulative database statistics on a system and session level are accessible using the `V$SYSSTAT` and `V$SESSTAT` views.

> **See Also:**
>
> *Oracle Database Reference* for information about the `V$SYSSTAT` and `V$SESSTAT` views

Interpreting Database Statistics

When initially examining performance data, you can formulate potential interpretations of the data by examining the database statistics. To ensure that your interpretation is accurate, cross-check with other data to establish if a statistic or event is truly relevant. Because foreground activities are tunable, it is recommended to first analyze the statistics from foreground activities before analyzing the statistics from background activities.

The following sections provide tips for interpreting the various types of database statistics to measure database performance:

- Using Hit Ratios

Using Hit Ratios

When tuning, it is common to compute a ratio that helps determine if a problem exists. Such ratios may include the buffer cache hit ratio, the soft-parse ratio, and the latch hit ratio. Do not use these ratios as definitive identifiers of whether a performance bottleneck exists. Instead, use them as indicators. To identify whether a performance bottleneck exists, examine other related performance data. For information about how to calculate the buffer cache hit ratio, see "Calculating the Buffer Cache Hit Ratio".

Using Wait Events with Timed Statistics

Setting `TIMED_STATISTICS` to `TRUE` at the instance level directs the database to gather wait time for events, in addition to available wait counts. This data is useful for comparing the total wait time for an event to the total elapsed time between the data collections. For example, if the wait event accounts for only 30 seconds out of a 2-hour period, then very little performance improvement can be gained by investigating this event, even if it is the highest ranked wait event when ordered by time waited. However, if the event accounts for 30 minutes of a 45-minute period, then the event is worth investigating. For information about wait events, see "Wait Events Statistics".

> **Note:**
>
> Timed statistics are automatically collected for the database if the initialization parameter `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`. If `STATISTICS_LEVEL` is set to `BASIC`, then you must set `TIMED_STATISTICS` to `TRUE` to enable collection of timed statistics. Note that setting `STATISTICS_LEVEL` to `BASIC` disables many automatic features and is not recommended.
>
> If you explicitly set `DB_CACHE_ADVICE`, `TIMED_STATISTICS`, or `TIMED_OS_STATISTICS`, either in the initialization parameter file or by using `ALTER_SYSTEM` or `ALTER SESSION`, then the explicitly set value overrides the value derived from `STATISTICS_LEVEL`.

> **See Also:**
>
> *Oracle Database Reference* for information about the `STATISTICS_LEVEL` initialization parameter

Using Wait Events without Timed Statistics

If `TIMED_STATISTICS` is set to `FALSE`, then the amount of time spent waiting for an event is not available. Therefore, it is only possible to order wait events by the number of times each event

was waited for. Although the events with the largest number of waits might indicate a potential bottleneck, they might not be the main bottleneck. This situation can happen when an event is waited for a large number of times, but the total time waited for that event is small. Conversely, an event with fewer waits might be a bigger bottleneck if the wait time accounts for a significant proportion of the total wait time. Without the wait times to use for comparison, it is difficult to determine whether a wait event is worth investigating.

Using Idle Wait Events

Oracle Database uses some wait events to indicate whether the Oracle server process is idle. Typically, these events are of no value when investigating performance problems, and should be ignored when examining wait events.

Comparing Database Statistics with Other Factors

When evaluating statistics, it is important to consider other factors that may influence whether the statistic is of value. Such factors may include the user load and hardware capability. Even an event that had a wait of 30 minutes in a 45-minute period might not be indicative of a performance problem if you discover that there were 2000 users on the system, and the host hardware was a 64-node computer.

Using Computed Statistics

When interpreting computed statistics (such as rates, statistics normalized over transactions, or ratios), verify the computed statistic with the actual statistic counts. This comparison can confirm whether the derived rates are really of interest because small statistic counts usually can discount an unusual ratio. For example, on initial examination, a soft-parse ratio of 50% generally indicates a potential area for tuning. If, however, there was only one hard parse and one soft parse during the data collection interval, then the soft-parse ratio would be 50%, even though the statistic counts show this is not impacting performance. In this case, the ratio is not important due to the low raw statistic counts.