Oracle 10g Self-Management Framework Internals: Exploring the Automatic Workload Repository

Open World
September 2005

ORACLE

Oracle 10g Self-Management Framework Internals: Exploring the Automatic Workload Repository

Oracle 10g Self-Management Framework Internals: Exploring the Automatic Workload Repository

INTRODUCTION

The Oracle 10g self-managing database effort necessitated the creation of a number of new classes of statistics and a repository in which to store them persistently for later analysis. The Automatic Workload Repository (AWR) was designed to address these requirements. AWR is primarily designed to provide input to higher-level components such as automatic tuning algorithms and advisors, but can also provide a wealth of information for the manual tuning process. This paper aims to give readers a better understanding of how AWR works and the nature of the data it collects. Though this information will be very useful to those engaged in the manual tuning process, it is our dual objective to instill confidence in our automated management features by detailing the infrastructure and information used to drive their decisions and recommendations.

AUTOMATIC WORKLOAD REPOSITORY BEHAVIOR

The Automatic Workload Repository is created and enabled by default in the Oracle Database 10g releases. Its persistent data is stored in a new, mandatory tablespace called SYSAUX. The following sections discuss the various behavior details and controls of the AWR.

AWR Snapshots

A fundamental aspect of the workload repository is that it collects and persists database performance data in a manner that enables historical performance analysis. The mechanism for this is the AWR snapshot. On a periodic basis, AWR takes a "snapshot" of the current statistic values stored in the database instance's memory and persists them to its tables residing in the SYSAUX tablespace. By default the length of the period is 60 minutes but can be modified anywhere from 10 minutes to any time length. The supplied PL/SQL package DBMS_WORKLOAD_REPOSITORY (see file *dbmsawr.sql*) provides the interface to adjust the snapshot period.

Each snapshot taken is assigned a sequence number value as its snapshot ID. The use of a sequence is important since a single snapshot is not very useful on its own. Consider that a snapshot is basically a capture of the current values of the database statistics, most of which have been monotonically increasing since the database

instance was started. To determine the statistical changes over a period of time, you need two snapshots to compute the value difference between the snapshots. For example, if you want to determine the number of database buffer gets on the system from noon to 1:00PM on a given day, you will need snapshots from both noon and 1:00PM. The difference in the value of the statistic between the snapshots gives you your answer.

AWR Snapshot Alignment is critical for period-to-period comparisons.

Snapshots are taken automatically by the database from a database background process called MMON. The actual snapshot processing is done from an MMON slave process typically named something like M001. To better support the ability to do comparisons between time periods across days, weeks, etc., the snapshots are aligned to the hour as much as possible. This means that if you want to compare the performance of the last hour to the same hour of the week last week you can assume that the snapshots from both weeks are aligned to each other by time of day.

Since MMON is a database background process, it minimizes the impact of performance data collection by accessing in-memory data directly from within the database. To further minimize the amount of data collected, snapshots collect only the most active SQL statements and database segments since the last snapshot was taken. The details of the collection of these statistic items will be discussed later, but we mention them here to bring up the point that some of the data in a snapshot is dependent on the time that the last snapshot was taken. This dependency between consecutive snapshots can be thought of as snapshot stream. There is only one snapshot stream in a database. This is an important concept to consider when making use of another AWR feature, the ability to manually create a snapshot. This feature is provided by the DBMS_WORKLOAD_REPOSITORY package. If you manually create a snapshot you will alter the meaning of the automatic snapshot taken immediately after the manual snapshot. For example, a manual snapshot taken at 11:30AM will mean that the automatic snapshot occurring at noon will now contain the top SQL as calculated from 11:30 to noon rather than the top SQL from 11:00AM to noon.

In an Oracle RAC environment, the AWR snapshots for each active instance are coordinated and given the same snapshot ID. A MMON slave process will create the snapshot from each instance, but the coordination will be done from a single instance. When working with snapshots from a RAC database you must specify both the instance ID and the snapshot ID.

AWR Purging

By default Oracle Database 10g is collecting AWR snapshot data every hour into the SYSAUX tablespace. To keep this data from growing uncontrolled, the AWR automatically takes care of purging data that has reached a certain age. The length of time in which AWR keeps data around before purging is referred to as the "snapshot retention period". By default, the retention period is set to seven days.

Oracle recommends that you consider increasing the retention period for AWR to one month. To estimate the space required to hold the data associated with a given snapshot retention period, there is a SQL*Plus script provided (*utlsyxsz.sql*). This script looks at the existing size of the database and asks a couple of questions about the anticipated workload size before giving a size recommendation.

Designate your peak periods by creating AWR Baselines for them.

AWR offers AWR baselines as a tool to override the purging process for important periods of time that will be valuable for future comparisons. The concept is simple, you just pick a series of snapshots (for example, 50 through 60) that correspond to the important time period and give it a name. As long as the baseline exists, the snapshot data will not be purged.

The AWR schema makes use of the Oracle partitioning feature to minimize the impact of snapshot purging. Performance data classes with potentially large amounts of snapshot data are stored into partitioned tables. These tables are partitioned around time (roughly one partition per day). Once a day, the AWR purge job is executed from an MMON slave process and drops partitions whose snapshots have all passed the snapshot retention time. It also creates new partitions for the next day's set of snapshots. The impact of this design is that the purge operation is very quick as compared to running large SQL delete statements.

AWR Snapshot Status

AWR is designed to have minimal impact on the running system, but it does require some amount of resources to perform its work. There is no guarantee that CPU, memory, etc will always be available and therefore AWR snapshots have to deal with failure situations. The good news is that AWR is tolerant of partial failures and is able to communicate the extent of the failure (what data is missing from the snapshot) to Oracle features that make use of the snapshot data. This information is also exposed to the user through dictionary views.

The dictionary view DBA_HIST_SNAPSHOTS stores metadata about each snapshot taken in the database. The column ERROR_COUNT gives the total number of AWR statistics tables we failed to populate during snapshot processing. Each failure is recorded into the DBA_HIST_SNAP_ERROR view. This contains the name of the table that we failed to collect as well as the error number that occurred.

If a snapshot has a zero value for ERROR_COUNT and has a valid END_INTERVAL_TIME, then it is considered complete. However, incomplete snapshots can often still be used for viewing or for tolerant tools such the AUTOMATIC DATABASE DIAGNOSTIC MONITOR (ADDM).

AWR STATISTICS CONCEPTS

Now that we have discussed the details of the AWR snapshot process, it is time to start looking at the specific statistics its collects. We start by describing the different conceptual types of statistics generated by the Oracle database and then

discuss some of the key statistics classes such as Wait Events and Active Session History (ASH).

Conceptual Statistic Types

AWR persists statistics by capturing the current values during a point in time snapshot. To analyze a particular period of time, you typically work with two snapshots and use the values captured at the end points to perform the analysis. The type of the statistic determines how you use the data to perform the analysis. For example, a counter type statistic such as "database buffer gets" can be used by simply subtracting the value in the first snapshot from the value in the last to get a total for the time period. What follows is a conceptual discussion of the types of statistics in the Oracle database, tailored specifically to help users analyze snapshot data.

The most basic type of statistic is a *counter* statistic. Counter statistics are monotonically increasing through time and only require the two end point values to calculate its value over a period of time.

The next type of statistic is a *value* statistic. A value statistic indicates the current outstanding amount of some item and can move up or down through time. For example, the total amount of private memory allocated to processes in the database system (PGA memory) is tracked by a value statistic. Taking the statistic value at the two end points of a snapshot period only gives the values at those exact points of time, but doesn't give much information about what happened between.

Another type of statistic is a *time* statistic. Time statistics represent the difference in values recorded against timers during an operation. Example timers are the wall clock timer and the CPU timer. An example of a timer statistic would be "DB CPU" which represents the change in the CPU timer on the system while database calls are active. Having the value of a time statistic at both end points of a time period reflects the amount of time taken during the time period.

Metrics give important minute-to-minute ratios of statistics relative to each other

The *Metric* statistic type is a composite type of the former three types. Metrics represents ratios of statistics over small intervals of time. For example, there is a metric defined for database buffer gets per database transaction over a one minute interval. The implementation of metrics requires periodic (frequency being the interval time) capture of statistic values into memory. For capture by AWR snapshots, these values must be stored in memory until the snapshot has a chance to capture them.

Finally, the *sampled* statistic type is based on taking periodic samples of statistic values or current state. The Active Session History (ASH) component is the Oracle database mechanism for generating these types of statistics. It samples the current state of sessions in the database in order to make time estimates on particular database activities. For example, sampled data from ASH can be used to estimate the time spent executing a particular SQL statement on the system. The advantage

is that sampled statistics are much cheaper to generate than measured statistics and are very accurate for operations that have enough samples.

AWR Statistic Aggregation

When a statistic is generated for an operation such as a database buffer get, the statistic can be charged to many entities such as the current session, the current SQL statement, or the current database segment. These are called statistics aggregation levels.

AWR supports the collection of statistics at a number of aggregation levels. The most pervasive is the system (instance) level. Other aggregation levels include service, SQL, and segment. For aggregation levels that have potentially large numbers of objects, AWR collects only the top N active objects on the system. This minimizes disk space while still collecting the most important activity on the system.

ASH creates a stream of session activity samples that can be analyzed along many dimensions.

Sampled statistics available through the Active Session History (ASH) facility support multiple aggregation levels. Each sample records active keys such as current session, SQL ID, service-module-action, etc. The samples can be analyzed along any of these aggregation dimensions.

STATISTICS CAPTURED BY AWR

AWR captures many classes of statistics during snapshots. This part of the paper will describe some of the most important statistics classes as well as some of the classes new to Oracle Database 10g.

Wait Event Statistics

A wait event in Oracle is a situation when a session puts itself to sleep to wait for some external event to happen. Examples are waiting for a shared lock to be released or waiting for an I/O request to complete. Typically any significant amount of time spent in wait events indicates a system problem unless the wait event is either an idle wait or a service wait. Idle wait events occur when a session is waiting for a work request to be issued. For example, while a session is waiting for the next request from the end user, it waits on an idle wait event. Service wait events are waits for the system to perform asynchronous operations such as I/O or process creation. In general, there has to be some amount time spent in these types of services. In Oracle Database 10g, wait events have been classified into wait classes. The classes have been specifically designed to direct tuning efforts. The wait class names with example wait events:

- Administration – backups, index rebuilds
- Application – row/table locks, user locks
- Cluster – RAC waits
- Commit – log file sync

- Concurrency – buffer busy, latches
- Configuration – free buffer waits, log buffer space
- Idle – rdbms ipc message, SMON timer
- Network – SQL*Net
- Scheduler – Resource Manager
- System I/O – db file write, log file write
- User I/O – reads, direct writes
- Other – rarely seen miscellaneous waits

AWR collects system level statistics for both individual wait events as well as for wait classes. These statistics include the elapsed time for a wait event and the number of occurrences of the wait event. They are viewable through the AWR dictionary view DBA_HIST_SYSTEM_EVENT. The class names and event names are available by joining with the DBA_HIST_EVENT_NAME view.

Time Model Statistics

While wait events typically show where the system is spending time waiting, the time model attempts to show where the system is wasting time doing activities such as reloading the SQL cache or parsing SQL statements that have parse errors. As the name implies, these statistics are made up of elapsed and CPU time statistics. AWR collects the system level Time Model statistics values for every snapshot. They are viewable through DBA_HIST_SYS_TIME_MODEL.

The Automatic Database Diagnostic Monitor (ADDM) uses both the wait model and time model to perform its system performance analysis. It detects problems by looking at where time is spent in the system. Once it has identified a problem area, it drills down to other classes of statistics to determine root causes and make appropriate recommendations for corrective actions.

A key statistic available from the time model is the "DB time" statistic. This statistic represents the total elapsed time spent servicing user requests in the database. The system level value for DB time can be much higher than the wall clock time that has passed because many sessions can be concurrently in a database call. DB time is used as the yardstick to measure other time statistics values. Given a time statistic such as a wait event, the ratio of its value to DB time gives the system impact of that wait event.

Active Session History (ASH)

ASH will flush its data directly to AWR if its buffer fills up before the next AWR snapshot.

The Active Session History is comprised of one-second samples of active sessions on the instance. A session is considered active if it is currently processing a database request (is in a database call). Generally, only a small set of connected sessions will be active at a given point in time. ASH samples are stored in a circular buffer in memory. This buffer is captured during AWR snapshots, but can be flushed onto disk in emergency situations if the buffer fills up before a snapshot is scheduled to run. In this situation, the flushed ASH data is associated to the next pending snapshot.

The ASH data that is collected during the AWR snapshot is only a tenth of what is captured into memory. This reduces the size of the ASH data persisted (ASH is one of the largest classes of statistics in the AWR snapshot) without losing a significant amount of accuracy. The in-memory data that is selected for writing is randomly chosen (sampling of the samples).

ADDM makes heavy use of the ASH data when trying to identify the root cause of a problem it has detected through either the time or wait models. Since ASH samples capture many dimensions of the activity occurring in the system, they can be used to find concentrations of a specific activity (such as a wait on a wait event). For example, if a particular database block is causing a lot of latch related waits, then ADDM will be able to identify the block from the concentration in the ASH samples.

The ASH samples contain many attributes. Some important ones are current session ID, current SQL statement ID, and current wait event ID. For a complete list you can reference the definition of the AWR view DBA_HIST_ACTIVE_SESS_HISTORY in the Oracle Database Reference Guide.

System Statistics

The database tracks a few hundred general counter, value, and time statistics aggregated at the system level. AWR collects these at each snapshot and makes them available through the DBA_HIST_SYSTAT view.

Top SQL Statistics

SQL statement statistics are cached in the shared pool along with the SQL statements. AWR collects statistics about SQL statements during snapshots so that this information can be used to identify SQL that is a potential target for tuning. ADDM uses the data for exactly this purpose; constructing the list of SQL with high concentrations of DB time and recommending that they be tuned. The other main consumer of this data, the Automatic SQL Tuning Advisor feature, makes use of the AWR collected data to reconstruct the environment in which high load SQL was parsed and executed.

When an AWR snapshot is run, the set of cached SQL statements may be very large and therefore it may consume a lot of space and other resources to capture

them. To reduce these overheads, AWR captures only the top SQL statements active over the snapshot period. Additionally, the changes since the last snapshot in the statistics it uses for selection criteria are available directly in the cached SQL statements. This avoids the need to access the previous snapshot's statistics in order to determine the top SQL.

AWR selects the top N SQL statements along five different selection criteria. The statistics used for the selection criteria are Elapsed Time, CPU Time, Parse Calls, Sharable Memory, and Version Count. For a description of these statistics, see the description of the V$SQLSTAT view in the Oracle Database Reference Guide.

By default, AWR will collect the top 30 SQL statements for each selection criteria during a snapshot. In Oracle Database 10g Release 2, the ability to adjust this setting was made available through the DBMS_WORKLOAD_REPOSITORY package. The value can be set to an explicit value (minimum is 30) or can be set to collect all SQL available in the SQL cursor cache.

AWR collects a variety of information for each SQL statement that it selects. It collects the execution plan for the SQL statement, the bind types bound for any bind variables, and a sample of bind values used in a previous execution of the SQL statement. There are more than 20 different counter, value and time statistics captured for each SQL statement.

Top Segment Statistics

AWR provides information about hot or problematic segments in the database.

Segment statistics are kept in the SGA about segments (tables, indexes, etc.) that are being accessed by the database instance. This data is very useful for identifying hot tables, missing indexes, row chaining, and a host of other data related issues. ADDM is one the primary consumers of this data. AWR will capture only the most active segments over the snapshot period. Similar to its top SQL collection, the snapshot processing will pick the top segments among five different statistical criteria. The criteria are the following:

- Logical Reads
- The sum of the statistics Row Lock Waits, ITL Waits, and Buffer Busy Waits
- The sum of the RAC statistics Global Cache Busy, CR Block Received, and Current Blocks Received
- The size change in the segment since the last snapshot
- The number times additional blocks were visited to retrieve rows (due to row chaining)

In addition to capturing the most active over the last snapshot period, AWR also attempts to periodically collect the statistics for segments that have been consistently active for a number of snapshot periods, but have never made the top active sets. This covers the case of segments whose use will only become significant over long time frames.

The segment statistics are available in the AWR view DBA_HIST_SEG_STAT. This view's description in the Oracle Database Reference Guide gives more details on these statistics.

Operating System Statistics

In Oracle Database 10g a new class of statistics was added to display operating system resource consumption. This data includes information about OS memory usage, CPU utilization, and in Oracle Database 10g Release 2 includes paging related information. ADDM is the primary consumer of this class of statistics. The actual statistics available vary a bit from operating system to operating system. The AWR snapshots capture this data (available in the V$OSSTAT view) on each snapshot. The historical information is available through the DBA_HIST_OSSTAT view.

Memory Statistics

AWR collects information about usage of both shared (SGA) and process private (PGA) memory. The information available in DBA_HIST_SGASTAT gives the current amount of memory allotted to different database components at snapshot time. Its statistics are value based. The view DBA_HIST_PGASTAT gives various statistics about the current total amount of memory being used by processes on the system, but also includes some counter statistics describing how many bytes have been processed by PGA memory operations.

Metrics

Metric data is produced at regular frequent intervals. These are typically every one minute, but some types of metrics are produced every 15 seconds. The volume of metric data is too large to store in detail during snapshot processing, but AWR does capture a summary of the key system level metrics over the snapshot period. This summary includes minimum, maximum, and standard deviation of the interval values over the snapshot period. The data is available in the view DBA_HIST_SYSMETRIC_SUMMARY.

There are cases where a select few metrics are written to AWR tables in non-summarized form and associated with snapshots. If a server generated alert is placed on a specific metric then any time that alert is fired all the metric data points will be flushed to AWR. Also, if you use Enterprise Manager's Dynamic Threshold feature, AWR will capture the detailed values for approximately ten different system metrics. In both cases, these metrics will be associated with snapshots and can be viewed in the views DBA_HIST_SESSMETRIC_HISTORY and DBA_SYSMETRIC_HISTORY.

Initialization Parameter Values

Init.ora parameters are part of the AWR snapshot captured data.

Each snapshot, AWR collects the current values of initialization parameters. Initialization parameter values and changes can have a profound impact on database performance and so this specific configuration information was deemed critical to collect along with the performance data.

VIEWING AWR DATA

AWR snapshot data can be viewed in a variety of ways. Statistics from AWR populate many of the Enterprise Manager (EM) screens. DBA_HIST views allow users to access the various classes of statistics directly through SQL. And of course various Oracle Database 10g advisors will process AWR performance data to output intelligent recommendations. If you choose to look directly at performance data in the database, Oracle provides some canned reports to make the data presentation more tailored to the task of performance tuning. These reports can be used to view performance data when EM or the DBA_HIST views are not available. This section of the paper will introduce the four AWR based reports and give some background on their intended use.

The AWR Report

The AWR report provides a detailed report of the statistics collected between two snapshots. The snapshots do not have to be consecutive and thus you can analyze any period of time bordered by AWR snapshots. The report can be generated and read through EM, or can be created in either text or html form through the DBMS_WORKLOAD_REPOSITORY package. There is also a SQL*Plus script provided with the Oracle release that can be used to generate the report (*awrrpt.sql*). Please refer to the Oracle Database Performance Tuning Guide for examples of how to manually create this report as well as the other three discussed in this section.

The AWR Compare Period Report

A very powerful technique for diagnosing performance problems is to compare a period of bad performance to a period with roughly the same workload having good performance. By isolating the differences between the detailed performance statistics of the period, you can narrow down the source of the performance problem. In Oracle Database 10g Release 2, AWR provides a report that presents the statistical differences between two sets of snapshot periods. For example, you can compare the performance of Monday 11:00AM to Noon this week to the same hour last week. The report presents the data ordered by the differences between the two periods rather than by the absolute values. Furthermore, it normalizes the data compared by the amount of time spent in the database (the "DB Time" statistic) so that periods of different lengths can be compared.

The AWR compare period report is available through EM and can also be created by directly calling a PL/SQL procedure in the

DBMS_WORKLOAD_REPOSITORY package or by using a supplied SQL*Plus script (*awrddrpt.sql*).

The ASH Report

The ASH report can be used to analyze short duration spikes in the system.

The Active Session History (ASH) sampled data is very useful for analyzing performance over short periods of time such as five minutes. Since ASH continually collects data into its buffer, it is keeping continual details of the system. In Oracle Database 10g Release 2, AWR has provided a report to target ASH data over any arbitrary amount of time. You are not confined to the snapshot interval. The report displays system activity over the specified time period along important dimensions such as time, SQL_ID, session, service, module, and action.

The ASH report is available through EM and can also be created by directly calling a PL/SQL procedure in the DBMS_WORKLOAD_REPOSITORY package or by using a supplied SQL*Plus script (*ashrpt.sql*).

The AWR SQL Detail Report

When AWR selects a SQL statement to capture during an AWR snapshot, it writes out more than just the SQL statistics. It captures the execution plan as well as bind information with sample values. To view the execution plan details about an individual SQL statement, AWR provides a SQL detail report that will display the plan information about one or more specific SQL statements over a series of AWR snapshots. It is useful for finding plan changes over the snapshot period or to see how the SQL statement(s) generally performed.

The AWR SQL Detail report is not yet available though EM, but can be manually created by directly calling a PL/SQL procedure in the DBMS_WORKLOAD_REPOSITORY package or by using a supplied SQL*Plus script (*awrsqrpt.sql*).

AWR and Performance

An in-depth performance study was conducted to measure the impact of automatic manageability features in Oracle Database 10g Release 1. The results of the study found that the performance impact was less than 2.5%. The design goal for the self-managing database effort was to be under 5%. Furthermore, the performance impact of AWR snapshots and of ASH sample collection was negligible. These asynchronous operations take little CPU and do everything possible to avoid contention with user workload by gathering data directly from SGA memory.

CONCLUSION

The Automatic Workload Repository was introduced into Oracle Database 10g to collect and persist statistics information about the Oracle database so that it can be used for performance analysis. It provides the data to most of the higher-level self-managing database features, and also manages its own data storage by purging old

data. It was primarily designed to automate the tuning process, but can also provide a wealth of data to be put to use by DBAs and database performance experts. With this in mind, AWR has provided simple and powerful mechanisms to extract and view the data. Now that you have a little better understanding of what is under the hood, we hope you will take AWR and its accompanying self-management features for a drive.

Oracle 10g Self-Management Framework Internals : Exploring the Automatic Workload Repository
September 2005
Author: Mark Ramacher
Contributing Authors: Gary Ngai, Mike Feng

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com